UNIVERSIDAD DE SEVILLA

DOCTORAL THESIS

Neuromorphic audio processing through real-time embedded spiking neural networks

---

Author:

Juan Pedro Domínguez Morales

Supervisors:

Dr. Ángel F. Jiménez Fernández
Dr. Manuel J. Domínguez Morales

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Robotics and Technology of Computers Lab.
Departamento de Arquitectura y Tecnología de Computadores

July, 2018

Declaration of Authorship

I, Juan Pedro Domínguez Morales, declare that this thesis, titled "Neuromorphic audio processing through real-time embedded spiking neural networks", and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UNIVERSIDAD DE SEVILLA

Abstract

Escuela Técnica Superior de Ingeniería Informática

Departamento de Arquitectura y Tecnología de Computadores

Doctor of Philosophy

Neuromorphic audio processing through real-time embedded spiking neural networks

by Juan Pedro Domínguez Morales

In this work novel speech recognition and audio processing systems based on a spiking artificial cochlea and neural networks are proposed and implemented. First, the biological behavior of the animal's auditory system is analyzed and studied, along with the classical mechanisms of audio signal processing for sound classification, including Deep Learning techniques. Based on these studies, novel audio processing and automatic audio signal recognition systems are proposed, using a bio-inspired auditory sensor as input. A desktop software tool called NAVIS (Neuromorphic Auditory VIsualizer) for post-processing the information obtained from spiking cochleae was implemented, allowing to analyze these data for further research.

Next, using a 4-chip SpiNNaker hardware platform and Spiking Neural Networks, a system is proposed for classifying different time-independent audio signals, making use of a Neuromorphic Auditory Sensor and frequency studies obtained with NAVIS. To prove the robustness and analyze the limitations of the system, the input audios were disturbed, simulating extreme noisy environments.

Deep Learning mechanisms, particularly Convolutional Neural Networks, are trained and used to differentiate between healthy persons and pathological patients by detecting murmurs from heart recordings after integrating the spike information from the signals using a neuromorphic auditory sensor.

Finally, a similar approach is used to train Spiking Convolutional Neural Networks for speech recognition tasks. A novel SCNN architecture for time-dependent signals classification is proposed, using a buffered layer that adapts

the information from a real-time input domain to a static domain. The system was deployed on a 48-chip SpiNNaker platform.

Finally, the performance and efficiency of these systems were evaluated, obtaining conclusions and proposing improvements for future works.

Acknowledgements

"We are what we watch, we are who we love, we are what we listen to; we become what we spend time with."

– Adam Nergal Darski

The project presented and described in this document has been, with no room for doubt, and by far one of the biggest challenges in my life. It all started at the end of 2015, and it has had too many ups and downs throughout the whole period of time. It has allowed me to fulfill some of my main life goals and dreams: working as a researcher and teaching at the university. I find those two activities fascinating and I would really love to focus my professional career on them.

I would have never been able to make it through this rollercoaster of emotions and succeed in this task if it had not been for all the given support throughout these years. Therefore, I would like to dedicate some words to different people that helped me to make this thesis possible, but also to memories, places and situations that happened during this period.

To begin with, my deepest gratitude goes to my supervisors Gabriel Jiménez Moreno, Ángel Francisco Jiménez Fernández and Manuel Jesús Domínguez Morales for their continuous support throughout the whole project. Thank you very much Gabriel for sharing part of your vast knowledge with me and the rest of the group every day, for helping me in this process and with all the documents while I was stressed, away or simply because you wanted to help. I cannot thank you enough for what you have done and for what you mean to me. Thank you very much Ángel for all the good advices and recommendations that you have gifted me and that you are still gifting me every single day. Thank you Manu for being such a great brother, teacher and inspiration. I owe my supervisors a lot. I couldn't have been luckier.

I would really like to thank the whole RTC research group for being like a second family to me. My gratitude goes to Saturnino Vicente Díaz, Elena Cerezuela Escudero, Anton Civit Balcells, José Luis Sevillano Ramos, Daniel Cascado Caballero, Manuel Rivas Pérez, Francisco Gómez Rodríguez, Lourdes Miró Amarante, Fernando Díaz del Río, and the rest of colleagues of the Robotics and Technology of Computers Lab. Special mention goes to Alejandro Linares Barranco, the new director of the department, for his advices and interesting comments during this project.

I would like to thank my colleagues and friends Antonio Ríos Navarro, Daniel Gutiérrez Galán and Ricardo Tapiador Morales for transmitting so much energy and will to work. You have been really helpful. We have shared "debate" meals, research trips, chats, barbecues, Rocket League games and innumerable experiences that I will never forget. I wish you the best. I would also like to thank the "new blood" of the group: Lourdes Durán López, Francisco Luna

Perejón and Isabel Amaya Rodríguez for reminding me who I am and cheering me up even if I wasn't down. Special thanks go to the electronic engineers team, consisting of Juanma and Alberto for their support and help. Alberto is the one with whom I shared not only my first months in the department but also many great experiences and chats.

Moreover, this work has been supported by the Spanish grant (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02/01), by the Andalusian Regional Excellence Research Project grant (with support from the European Regional Development Fund) MINERVA (P12-TIC-1300) and by the Spanish government grant (with support from the European Regional Development Fund) COFNET (TEC2016-77785-P). I would also like to thank the Spanish Ministry of Education, Culture and Sport for my Formación de Personal Universitario grant, which helped me to have full dedication in this thesis. My deepest gratitude for this support.

During this period, I also had the opportunity to visit foreign institutions. There, I had the chance to meet extraordinary researchers that helped me to become the researcher I am today. Among these people, I will never forget Qian Liu, Robert James, Luis Plana, Simon Davidson, Steve Furber, Oliver Rhodes and Garibaldi Pineda García, from the APT Research Group in the University of Manchester (United Kingdom). I will also never forget Waleed el-Shobaki for being such a great person and host during my stay in Manchester and Jesko Meißel for all the good times that we had during that stay.

I have also had the opportunity to attend international conferences and workshops in many different places: Waterloo (Canada), Manchester (United Kingdom), Krakow (Poland), Barcelona (Spain), Cádiz (Spain), Alghero (Italy), Baltimore (United States), Florence (Italy) and Rio de Janeiro (Brazil), where I have met incredible and inspiring people that I have the privilege of calling them friends, like Frank Faries and his wife Tracey, Diederik Paul Moeys and Tim Walther.

Moreover, I am extremely lucky to have my wonderful parents, Manuel Domínguez Roldán and Isidora Morales González, my brother Manuel Jesús Domínguez Morales and his wife Rosselyn Rojas. I cannot thank you enough. I want to thank my family for all their support and love.

Special thanks go to my friends: to the ones I had, to the ones I have and to the ones I will have. You know who you are and you are an essential element in my life.

To all of them, once again, thank you.

Contents

List of Figures

List of Tables

List of Abbreviations

AER	**A**dress **E**vent **R**epresentation
AGC	**A**utomatic **G**ain **C**ontrol
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
ANSI	**A**merican **N**ational **S**tandards **I**nstitute
APT	**A**dvanced **P**rocessor **T**echnologies
ASIC	**A**pplication-**S**pecific **I**ntegrated **C**ircuit
ASR	**A**utomatic **S**peech **R**ecognition
CAR	**C**ascade of **A**symmetric **R**esonators
CFB	**C**ascade **F**ilter **B**ank
CNN	**C**onvolutional **N**eural **N**etwork
CSV	**C**omma-**S**eparated **V**alues
CVD	**C**ardio-**V**ascular **D**isease
DAPGF	**D**ifferentiated **A**ll-**P**ole **G**ammatone **F**ilter
DL	**D**eep **L**earning
DNN	**D**eep **N**eural **N**etwork
DVS	**D**ynamic **V**ision **S**ensor
EPSP	**E**xcitatory **P**ost-**S**ynaptic **P**otential
ERB	**E**quivalent **R**ectangular **B**andwidth
FAC	**F**ast-**A**cting **C**ompression
FFT	**F**ast-**F**ourier **T**ransform
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray
GALS	**G**lobally **A**synchronous **L**ocally **S**ynchronous
GUI	**G**raphical **U**ser **I**nterface
HWR	**H**alf-**W**ave **R**ectifier
IHC	**I**nner **H**air **C**ells
IPSP	**I**nhibitory **P**ost-**S**ynaptic **P**otential
ILD	**I**nteraural **L**evel **D**ifference
ISI	**I**nter-**S**pike **I**nterval
ITD	**I**nteraural **T**ime **D**ifference
LIF	**L**eaky **I**ntegrate-**A**nd-**F**ire
LINQ	**L**anguage **I**ntegrated **Q**uery
NAS	**N**euromorphic **A**uditory **S**ensor
NAVIS	**N**euromorphic **A**uditory **VIS**ualizer
NLP	**N**atural **L**anguage **P**rocessing
NN	**N**eural **N**etwork

NSP	**N**oisy **S**oft**P**lus
OHC	**O**utter **H**air **C**ells
OZGF	**O**ne-**Z**erod **G**ammatone **F**ilter
PAF	**P**arametric **A**ctivation **F**unction
PFM	**P**ulse **F**requency **M**odulation
RNN	**R**ecurrent **N**eural **N**etwork
RTC	**R**obotics and **C**omputer **T**echnology
SCNN	**S**piking **C**onvolutional **N**eural **N**etwork
SH&F	**S**pike **H**old **&** **F**ire
SLPF	**S**pike **L**ow **P**ass **F**ilter
SNN	**S**piking **N**eural **N**etwork
SNR	**S**ignal-to-**N**oise **R**atio
SPL	**S**ound **P**ressure **L**evel
SSP	**S**pike **S**ignal **P**rocessing
STDP	**S**pike-**T**iming-**D**ependent **P**lasticity
VLSI	**V**ery **L**arge **S**cale **I**ntegration

To my family and friends.

Part I

Thesis

Chapter 1

Introduction

"It's overwhelming just to think I was so blind.
The answers I've been searching for are already here."
– Justin Bonitz

Human beings have tried along their history to artificially solve different problems that had been encountered in their path and that were already solved in nature, in order to use the same mechanisms to progress and facilitate the execution of dangerous or heavy tasks. The inquisitiveness for the unknown has led to the development of smart solutions to survive and evolve in environments to which humans were not adapted.

New research fields have been opened regarding the way to process the information that we obtain from our surrounding environment through different sensors. Some applications have the ability to provide autonomy to artificial devices, such as machines in order to automate processes without human intervention and the development of new devices capable of solving problems that are increasingly complex with a higher precision and in a more efficient way.

These advances, for instance, were clearly seen in factories, with the goal of developing robots to perform dangerous, high-precision and repetitive tasks by providing them with the necessary intelligence, which would not be profitable or possible to carry out if a person performs them. Currently, and increasingly often, the technological devices that we buy and use in our daily basis have some kind of intelligence that helps us achieve many of the tasks and actions that previously they could not perform. The way in which our smartphone is able to cancel part of the ambient noise during a call in order to improve the quality of the conversation, or how cochlear implants are able to adjust its gain and filters to adapt the input to specific environments, improving the customer's quality of life, are a couple of examples, from among others.

A new learning and automatic processing mechanism inspired by the behavior of the nervous system was born in the recent history of artificial intelligence, known as artificial neural networks (ANN). It is an interconnected system of neurons that process the information together in order to produce

stimuli and generate a specific output. These algorithms have raised in popularity and many applications can be found in which neural networks are involved. Some of the most common uses for these networks are pattern matching and pattern recognition based on the data used as input to the network, generating a specific output after the information is processed. Based on this concept, many different types of neural networks specialized in specific tasks have appeared. Some examples are convolutional neural networks, known in the literature as CNN, which have proved to be ideal for extracting semantic information from the input data, and spiking neural networks, known as SNN, which offer a bio-inspired approach when processing the information.

A new line of research called Neuromorphic Engineering was born at the end of the 80s in relation to these matters and problems. It focuses on studying biological and inner systems of the human neural processing (Maher et al., 1989; Mead and Mahowald, 1988), aiming to achieve analog and digital systems able to mimic or produce the same operating patterns of the neurons in the human brain. New sensors and bio-inspired systems are proposed in this line of research, targeting at becoming high-speed data transfer and processing solutions, along with understanding the behavior of the brain by mimicking part of it. In the recent literature, it can be found: vision sensors (Lichtsteiner et al., 2008; Serrano-Gotarredona and Linares-Barranco, 2013), auditory sensors (Chan et al., 2007; Jiménez-Fernández et al., 2017; Yang et al., 2016; Xu et al., 2018), motor control systems (Jimenez-Fernandez et al., 2012; Perez-Peña et al., 2013; Gómez-Rodríguez et al., 2016), spiking neural networks (Stromatias et al., 2015; Sen-Bhattacharya et al., 2018), and sensor fusion (Pearson et al., 1988; Chan et al., 2012; Ursino et al., 2016), among many others.

This work is focused on neuro-inspired processing[1] of acoustic signals, aiming to obtain an efficient, automatic, noise-immune sound recognition system. This kind of processing is usually done in two different phases: to extract characteristic patterns to describe the acoustic signal, and to identify the studied sound by comparing the characteristics that were extracted in the previous step with reference models. These two sound recognition phases are already present in nature in real-time, even in changing environments with background noise and tough conditions. For the first part, we have used a neuromorphic auditory sensor (NAS) (Jiménez-Fernández et al., 2017), which mimics the functionality of the biological cochlea and how the information is coded in the auditory nerve. This system processes the information by means of narrow pulses, using the time between them as a modulation mechanism (Pulse Frequency Modulation or PFM). A novel software tool for post-processing the output information of a NAS has been developed in this work in order to study and analyze the frequency components of specific sets of audio samples. Regarding the sound identification phase, this research has focused on automatic pattern recognition using different types of NNs. We have developed three different systems: the first one is based

[1] It is based on how the nervous system codes and processes the information.

on SNNs, the second one on CNNs and the third one uses a combination of both spiking and convolutional neural networks, SCNN. These classifiers are able to extract specific patterns from the frequency components of the information provided by the neuromorphic sensor.

The most common tasks that can be found in the literature regarding neuromorphic sound recognition systems are: locating the sound source (Chan et al., 2007; Schaik et al., 2009; Chan et al., 2012; Cerezuela-Escudero et al., 2018), determining the nature of the sound (Nielsen et al., 2006; Ding and Zhang, 2007; Jäckel et al., 2010) and understanding the sound meaning (Kim et al., 2009; Barbancho et al., 2012; Miró Amarante, 2013; Jesus Guerrero-Turrubiates et al., 2014).

1.1 Motivation

Sound recognition systems are needed in a wide variety of different applications, such as music transcription, speech recognition, "speech-to-text" transcription, in-car systems and voice commands control, language learning, helping people with disabilities and even for medical and healthcare applications like cochlear implants or specific diseases diagnosis based on audio signals.

Automatic speech recognition (ASR) is a difficult task because of the acoustic signals variability. Under favorable conditions, the audio signal recognition could perform well, but things change under real conditions. Changes in the acoustic (noise, reverb and echo) or electric (noise and signal distortions) environments are some of the main factors that are present in a real scenario. Another important factor that makes this a complex task is the need for a wide dynamic range in order to discriminate sounds by their frequency and temporal structure. Moreover, unlike images, where the information is given in the spatial domain (the relation between adjacent pixels), the main information of audio signals is given in the time domain, adding another element to the complexity of the recognition system.

Regarding the living beings' nervous and sensory system, it is still not known how they are capable of achieving sound recognition with such high efficiency by using, almost exclusively, information that is based on spiking signals (Engineer et al., 2008). Bio-inspired solutions can overcome the problems related to sound recognition that were mentioned previously, mainly because biological cochleae provide a wide dynamic range and immunity to noise, while also spiking neural networks achieve good results for recognizing frequency-based patterns.

Sound recognition can be achieved with current technology by digital hardware and software platforms, analog platforms or even a combination of both. In this work we use bio-inspired spiking signals based on Pulse Frequency

Modulation (PFM), commonly used in the neuromorphic engineering field, to solve problems related to sound recognition tasks. The auditory sensor used to obtain the spike-based information decomposes the input sound signal into its frequency bands. Obviously, we are not going to exactly mimic the nervous system of living beings; however, by using artificial spiking neural systems, we can better understand how some features related to this kind of processing are performed in nature. On the other hand, new mechanisms and systems based on a neuro-inspired approach could provide specific technological advantages to diverse applications. In short, these are the two main motivations that have led scientific research in general: understanding nature and using that knowledge to improve technological advancement. The results presented in this work are not only isolated results to solve specific problems, but also a way to proceed for different sound recognition tasks and applications and a way to understand how biology performs this kind of processing.

On the personal side, it is important to highlight the author's inquisitiveness for researching new ways in which spiking information obtained from audio signals could be processed and trained for building bio-inspired classification systems, which could be used for automatic robot navigation and decision-making tasks based on speech recognition, among others, along with the development of new software tools for analyzing and postprocessing this kind of spike-based information.

This thesis is part of the research career of the Robotics and Computer Technology group (RTC, TEP-108), to which the author belongs. This work is focused on and aligned with different tasks that are part of national research projects, which have served as funding, along with the Spanish Ministry of Education, Culture and Sports, thanks to a Formación de Profesorado Universitario scholarship (2014, BOE-A-2014-13539). These projects are:

- BIOSENSE national project: Sistema bio-inspirado de fusión sensorial y procesamiento neuro-cortical basado en eventos. Aplicaciones de alta velocidad y bajo coste en robótica y automoción (TEC2012-37868-C04-02).
- COFNET national project: Sistema Cognitivo de Fusión Sensorial de Visión y Audio por Eventos (TEC2016-77785-P).
- MINERVA excellence project: Mota-Infraestructura de Sensado y Transmisión Inalámbrica para la Observación y Análisis de la Pauta de Animales Salvajes o en Semilibertad (P12-TIC-1300).

A detailed introduction about the main scientific fields that this thesis comprises is presented in the following sections, starting with bio-inspired systems, continuing with the auditory system and audio processing, and ending with deep learning and a hardware computing system that was used in this work to model bio-inspired networks.

1.2 Neuro-inspired systems

Since the inception of life on Earth, living beings have spread across the planet to colonize different habitats. One of the keys to this life expansion is the capacity of adaptation of organisms, who develop by nature the necessary traits to survive a specific environment. Moreover, because of the high diversity of natural habitats, organisms have had to specialize in surviving in their closest environment and, therefore, this has produced a very rich diversity of species. These particular intrinsic characteristics are coded within the genetic code of each species, which has been modified and molded since the origin of life through evolutionary processes. Thanks to evolution, a great variety of species can be found in nature. Thus, many efficient solutions to the adaptation problem for organisms in their environment have been achieved.

Throughout history, engineers have observed and taken inspiration from how different problems are solved in nature in order to solve other complex problems. This marks the inception of bio-inspired systems, which can be found around us more and more often. Some bio-inspired systems are: drones, submarines, planes, humanoid robots and cochlear implants, among others.

In the last decades, a big revolution has taken place in both industry and our daily lives, thanks to the appearance of computation and robotic systems. Industry has progressively incorporated the use of robots to perform complex tasks where high precision and repeatability are needed or a lot of effort has to be made. However, this kind of robots are programmed to perform a very limited number of tasks in a controlled environment, with a high power consumption and without any kind of ability to learn. On the other hand, animals are able to navigate freely within the environment and they can also provide themselves with energy, learn, socialize with other animals, develop a unique personality, work together to achieve a common goal, etc.; that is, animals are capable of developing social and cognitive abilities.

Most robots are currently commanded by algorithmic behaviors that are processed by computer-based systems. In the last years, computers have evolved at a very high rate, achieving a vast computational capacity, as was predicted by Moore's Law (Moore, 2006). Even though computers have evolved considerably in the last years, robots have not experienced such a huge evolution, improving at a much lower rate. It is at this point when we should wonder about the possibility of modelling the behavior of a living being or one of their functionalities with the necessary fidelity to algorithmically code this behavior in a program and load it into the robot. If this approach was possible, could this algorithm be executed in a computer in a relatively low time and with at least a minimum reliability level? Possible answers to this approach could lead to new approaches and questions, and, among all of them, we should wonder if current computer-based systems

are the most appropriate ones to provide robots with advanced cognitive abilities (Penrose and Mermin, 1990).

The answer to each of these questions is currently unknown, although one thing is clear: we should take into account how the brain processes the information and try to mimic the "controlling system" (central nervous system) of living beings themselves by using reverse engineering techniques. From this idea, the concept of neuro-inspired systems was born. These systems are a subset of bio-inspired systems, which try to solve common engineering problems by using systems that are based on how the nervous system encodes and processes the information. This research area is increasing in popularity thanks to the work of neuromorphic engineers. Processors have been at a standstill for some years, and thus, a good alternative is to explore how the brain works and processes the information using specific purpose systems instead of general purpose systems.

This section describes the concept of neuromorphic engineering, its brief history and the principles by which it is governed.

1.2.1 Neuromorphic engineering

The term neuromorphic engineering was first used by Carver Mead in the California Institute of Technology (Caltech) in the late 80s, with the initial goal of mimicking the behavior of neurons in nervous systems by means of VLSI analog circuits or aVLSI (Liu et al., 2002). His contribution consisted in understanding biological neural systems through silicon implementations, which has inspired the field of analog neuromorphic circuits design (Mead, 1989). However, the research interests of neuromorphic engineers have grown in the last years, focusing on using also digital circuits instead of only analog circuits to model the behavior of a neural system. To achieve this goal, it is important to understand how nature has been able to create complex, noise-robust and very efficient neural architectures that are also capable of learning. Neuromorphic engineering is defined as a research field that is dedicated to design and develop artificial computing systems whose physical properties, structures or representation of the information are based on the biological nervous system. Researchers from many different research fields take a place within neuromorphic engineering: physicists, mathematicians, physicians, biologists, engineers and even psychologists.

Since its inception in Caltech and Johns Hopkins University, this community has grown considerably. Only a few prototypes of neuromorphic systems could be found in some laboratories during the first years since its inception, but in the last 20 years many neuromorphic platforms have appeared. Some of them are able to deploy and simulate neural network models with thousands of neurons (Furber et al., 2014; Indiveri et al., 2006). Moreover, we can find neuromorphic sensors with a high dynamic range which can be used for high performance

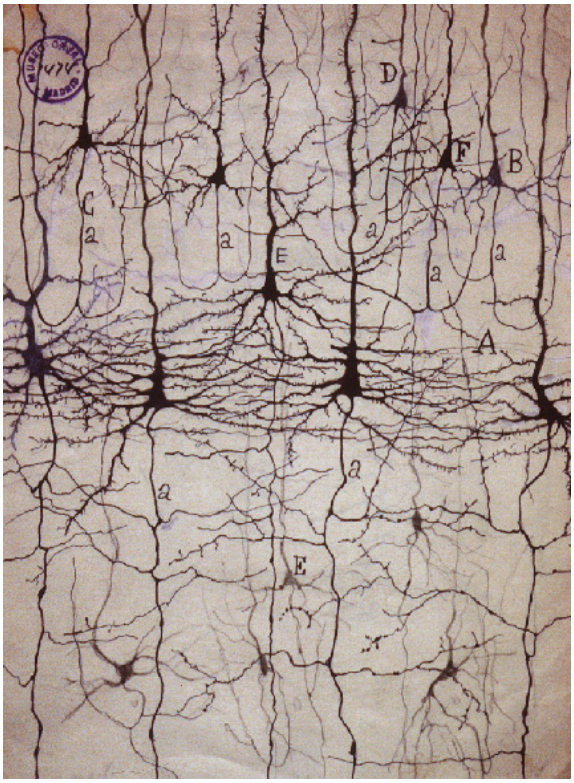


FIGURE 1.1: Reproduction of an original Ramon y Cajal drawing that shows a few neurons in the mammalian cortex that he observed under the microscope.

applications (Chan et al., 2007; Lichtsteiner et al., 2008; Jiménez-Fernández et al., 2017; Serrano-Gotarredona and Linares-Barranco, 2013).

Currently, two international workshops in the field of neuromorphic engineering take place every year: one in Telluride (United States) and another one in Capo Caccia (Italy). These two workshops are a meeting point for researchers all over the world, in which they have the opportunity to share their last advances and developments in this field with the rest of the community, along with collaborating and working with people from other research groups and sharing their neuro-inspired systems to design specific systems for processing spiking information.

In 1906, the Spanish scientist Santiago Ramón y Cajal received a Nobel Prize in Physiology or Medicine in recognition of his work on the structure of the nervous system, discovering that the brain consists of a set of independent and interconnected cells, the neurons. His studies were possible thanks to the

advances in staining methods and microscopes. A reproduction of one of Ramon y Cajal's drawings is shown in Fig. 1.1.

Three different parts can be distinguished in a neuron: dendrites, soma and axon. A scheme of the neuron and its three main parts can be seen in Fig. 1.2. Dendrites are the "inputs" of the neuron and they collect information from other neurons, which is then sent to the soma. The soma is the "central processing unit" and it performs a non-linear processing of the received information. The information that is received and processed modifies the potential that a neuron has and, if a specific threshold is reached, a signal is generated, which will be sent through the axon (the "output" of the neuron) to other neurons that are connected to this one.

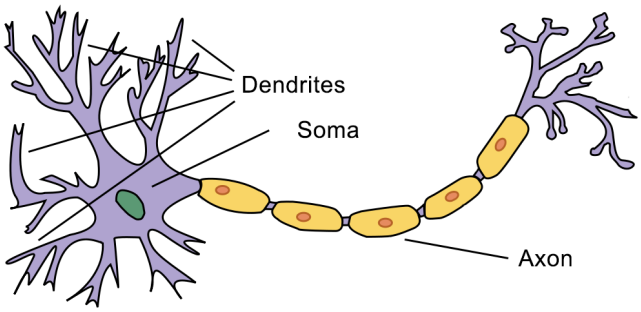


FIGURE 1.2: Basic scheme of a neuron.

The connection between two neurons is known as the synapse (Johnston and Wu, 1994). Neurons have very complex physiological characteristics. In 1939, Hodgkin and Huxley (Hodgkin and Huxley, 1939) analyzed the electrical behavior of an isolated neuron, studying how its sodium and potassium channels behaved, which granted them the Nobel Prize in Physiology or Medicine in 1963, demonstrating that neurons represent, communicate and process the information by means of small electric pulses in time, known as action potentials or spikes. The neuron that sends a spike through a synapse is called pre-synaptic neuron, whereas the one that receives the spike is called post-synaptic neuron.

Neurons have a resting potential of about -70mV. If the opening of the ion channel results in a net gain of positive charge across the membrane, the latter membrane is said to be depolarized, as the potential comes closer to zero. This process is called excitatory post-synaptic potential (EPSP), as it brings the neuron's potential closer to its firing threshold (about -55 mV). On the other hand, if the opening of the ion channel results in a net gain of negative charge, this moves the potential further from zero and is referred to as hyperpolarization. This process is called inhibitory post-synaptic potential (IPSP), as it changes the charge across the membrane to be further from the firing threshold. If

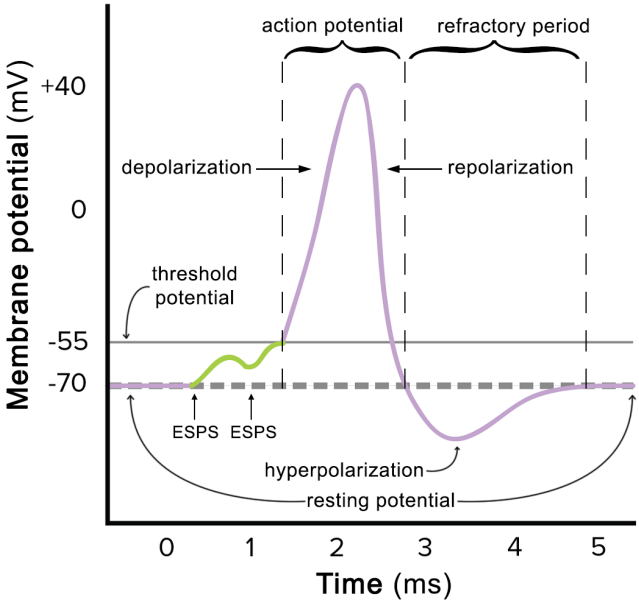


FIGURE 1.3: Diagram of a spike generated by a neuron.

the membrane potential of the neuron reaches the threshold, the neuron will depolarize abruptly, generating a spike and transmitting a nervous impulse. After firing[2] the spike and the depolarization of the neuron, it will polarize up to the point of hyperpolarizing, being unable to emit a new spike until a specific time period, known as refractory period, has passed. Fig. 1.3. shows the diagram of a spike generated by a neuron.

One of the key aspects of neural processing mechanisms is the hypothesis on how neurons represent the information (how the information is encoded into spikes). Horace Barlow proposed different models (Barlow, 1961). One of them, which is widely accepted within the neuromorphic engineering community, proposes that the information is encoded in the frequency of the spikes using PFM (Maass and Bishop, 2001; Westerman et al., 1997; Shepherd, 2003). This way, the information can be encoded without the need of performing a temporal discretization of the information (Hynna and Boahen, 2001; Fujii et al., 1996). Other ways to encode the information are through the interspike interval (ISI) (Indiveri et al., 2006), or through the reset time, where the most important events are the ones that have been emitted first (Thorpe et al., 2010).

The spiking representation of the information is very efficient from different points of view: its simplicity, reducing the number of communication channels

[2]In the neuromorphic engineering field, firing a spike is synonym to emitting a spike.

needed to transmit spikes, and its continuity, providing a continuous flow of information in time, instead of in a discrete way. Therefore, spiking representation provides a minimization in the number of channels needed for communication, allowing a high interconnectivity rate between neurons, and, since the information is not sampled, it avoids transmitting redundant information and saturating the communication channels in an unnecessary way by only transmitting spikes when they are needed.

This representation is efficient not only in the communication and connectivity point of view, but it is also very robust to noise: the information is encoded in the time between two consecutive spikes (ISI), where it is only important if a spike exists or not. However, analog signals are completely defenseless against external perturbations.

In most animals, the brain is the main part of the central nervous system. It is located in the head, close to the most important sensory organs and it is protected by the cranium. The human brain is extremely complex and it is estimated to have between 15 and 33 trillion neurons, where one single neuron can be connected to another 10 thousand. Neural structures are grouped in layers, whose goal is to process part of the information that is obtained in the input. Each layer has a specific functionality (Shadlen and Newsome, 1994; Rakic, 1988).

TABLE 1.1: Qualitative comparative analysis between a computer and a neural system.

Computer	Neural system
High speed global clock signal.	Asynchronous, without a global clock signal.
Deterministic behavior.	Stochastic behavior.
High resolution information sampled at a constant rate.	Low resolution, but adaptive. No sampling rate; the information is encoded within the frequency of the spikes.
Centralized computing, or slightly distributed.	Each neuron processes a small part of the information. The processing is highly distributed and massively parallel.
It needs memory for the algorithm and to store the data.	The information is encoded within the flow of spikes. The morphological characteristics and the interconnections of each neuron is the algorithm itself.

Table 1.1 shows a generalized qualitative comparison between how a computer and a neural system work. The first difference that can be appreciated is that a computer is controlled by a global clock signal, which makes it react continuously every clock cycle. However, for neurons the processing is completely asynchronous and they do not have any synchronization mechanism. Moreover, the computer is a completely deterministic element that, after a sequence of arithmetic and logic operations, knows in which state it should be, while neurons respond to a stochastic model, depending on their reaction to

their dynamic probabilistic models. Computer-based systems work with high resolution information that is sampled at a constant rate, whereas neurons, once again, are the exact opposite: although the resolution is not that high, they are able to adapt the characteristics of the information to improve its representation. In current computational systems, the processing is centralized (e.g. a personal computer), in contrast to the way in which neurons process the information: each neuron processes a small part of the information, without depending on other neurons, implementing a massively parallel model. Computers need memory units to store the instructions that are going to be executed and also for the initial, halfway and final data, while neural systems do not need memory for any of these purposes due to the fact that the "neural algorithm" that they perform is encoded in the connection between different neurons and in the physiological characteristics of them. This is, the information processing is in the spike stream itself.

1.2.2 Address-Event Representation

Most neuromorphic systems consist of one or more neuromorphic sensors and a set of spiking neural network layers to process the information provided by the sensor, trying to mimic the interconnection that biological neurons have. Unfortunately, based on the physical limitations in terms of connectivity and taking into account the high density of connections in the brain, where each of the $10^5/\text{mm}^3$ neurons could be connected to other 10 thousand generating a density of connections up to $4\text{km}/\text{mm}^3$ (Braitenberg and Schüz, 1998), it is not possible to implement such connectivity in a VLSI system. Nevertheless, a neuron presents a firing rate that ranges between 1-10 Hz, scaling up to kHz or MHz if many hundreds are combined together, meaning that state-of-the-art electronic circuits are much faster than biological neurons. To solve the connectivity problem and based on this time difference and the high-bandwidth capacity of VLSI systems, a mechanism is proposed that multiplexes the information in time for a set of neurons in a single communication channel, where each neuron is identified with a unique address. This method for representing the information is known as Address Event Representation (AER), which was proposed for the first time in 1991 (Sivilotti, 1991; Lazzaro et al., 1993; Lazzaro and Wawrzynek, 1995; Boahen, 2000). It is an event-driven communication protocol used originally for transferring spikes (action potentials) between neurons in VLSI implementations (Mahowald, 1992). However, it is also useful for transmitting and receiving large amounts of rate-coded information through a channel with smaller bandwidth (digital asynchronous bus). Thus, it is an event-driven asynchronous and digital multiplexing technique.

The main functionality of AER circuits is to provide multiplexing / demultiplexing mechanisms for spikes that are generated by / sent to a set of neurons. Fig. 1.4 schematically shows the transmission of information between

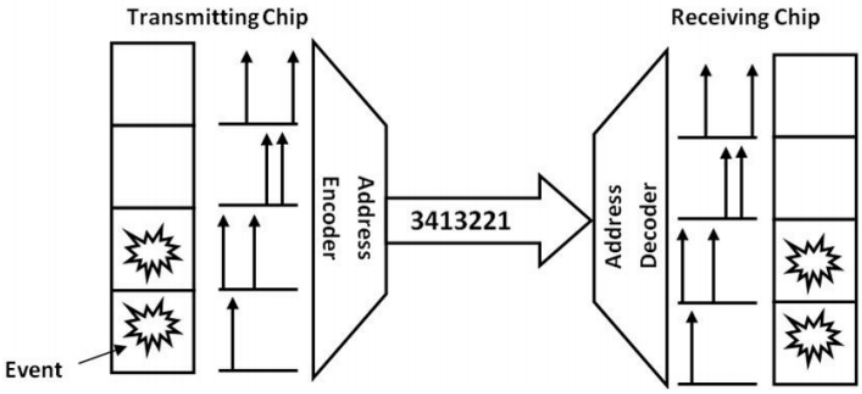


FIGURE 1.4: Transmission of spiking information using AER representation.

two neuromorphic chips using the AER protocol. This protocol uses a shared multiplexed high-speed bus (AER bus) for transmitting the spikes that are fired by the neurons on a chip. Each neuron is identified with a unique address. Every time that a neuron spikes, and thanks to an arbiter circuit, the address of that neuron will be placed in the AER bus, generating an AER event. This way, each of the asynchronous spikes will be encoded and multiplexed by the AER circuit in the bus in the same order as they were generated. The timestamp in which the address of the neuron is generated corresponds to the timestamp in which it was fired plus a small delay caused by the codification process. These coding circuits use a specific arbitrating logic to handle the transmission of multiple spikes simultaneously from different neurons. Many different AER encoders can be found in the literature, based on the mechanism used to solve the conflict of multiple simultaneous spikes and how the addresses are encoded (Cerezuela-Escudero et al., 2013). Each of these options have their own advantages and disadvantages depending on the neuromorphic system used. A comprehensive study of these mechanism can be found in (Liu et al., 2015).

The address of the neuron that produced the AER event is decoded in the receiver chip as soon as it arrives using an asynchronous decoder, sending the initial spike to the corresponding neuron. This way, neurons of the transmitter and the receiver are virtually connected through the AER bus using the AER protocol. If the delay between adjacent spikes in the input is high enough, the AER decoder will send the spikes to the corresponding neurons with a delay that corresponds to the time that the spike takes to reach the decoder plus a small delay caused by the decoding circuit.

The AER protocol is a 4-phase handshake protocol between the transmitter and the receiver that guarantees the synchronization between both chips. The

transmitter starts the communication process with a request. Then, the second chip, the receiver, answers to this request with an acknowledgement. To conclude the transmission, the transmitter removes the request and the receiver does the same with the acknowledgement, resetting the system to its initial condition. Both parts of the communication are not active until the transmitter starts a new request. A new transmission depends on the neurons of the transmitter chip that try to send and AER event. Thus, AER is a data-driven protocol. In this asynchronous protocol, the activity within the communication bus depends on the data transmissions, which is in contrast with classical discrete and periodic systems.

In this thesis, the AER protocol is the basis of the transmission and processing of the auditory information. The AER protocol used in this work was proposed in the European research project CAVIAR: Convolution AER Vision Architecture for Real-Time (IST2001-34124) (Serrano-Gotarredona et al., 2009). The characteristics of the cable and the connector for implementing this AER protocol are described in the AER specification document (Berge and Hafliger, 2007).

1.2.3 Software tools for processing neuromorphic information

With the increasing number of neuromorphic hardware including sensors and systems for deploying and running SNNs in real time, a proper set of software tools to process spiking information have become very useful for various reasons:

- Debugging neuromorphic sensors is a challenge due to the fact that the output information that they provide is encoded in the time between two consecutive spikes (inter-spike interval or ISI). Thus, tools for analyzing the information from the output of these sensors are required.
- Users could only want to obtain the spiking information from the sensor and process it using particular algorithms in order to extract useful data from it and perform some sort of classification or study. For this purpose, software tools able to log the information from the sensor to the computer using common interfaces and processing it both in real time or after recording it to the hard drive are needed.
- In order to train a neural network with the output information from a neuromorphic sensor, a software application is needed to generate a dataset out of the logged information, extracting the useful components and saving them in a standard format.
- Prior to deploying a SNN implementation on a neuromorphic hardware system, it is commonly built, trained and tested on a simulator in order to verify that the specific model to be implemented behaves as expected for a generalized set of input samples. Several libraries like PyNN (Davison et al., 2009) and Nengo (Bekolay et al., 2014) allow us to build complex

spiking networks, and simulators like NEST (Gewaltig and Diesmann, 2007), Brian (Goodman and Brette, 2008) and NEURON (Hines and Carnevale, 2001) can simulate those models before deploying the network in a hardware platform.

Some of these software tools for processing neuromorphic information are presented in the following chapters.

1.2.3.1 jAER

jAER (Delbruck, 2008) is an open-source (under GNU Lesser General Public License v2.1) framework for PCs for visualization of real-time or recorded event-based data[3], and rapid development of real-time event-based algorithms and applications built in Java. jAER consists of an application called "jAERViewer" that allows to plug in any AER device with USB interface and perform different functionalities with it, e.g. view the events coming from the device in real time, log (record) them to disk, play a logged AER stream back and process the events using different filters.

There are many AER devices compatible with jAER, from dynamic vision and audio sensors to AER monitor/sequencer boards, along with a servo motor controller, among others.

The events are produced by sensors asynchronously and timestamped (with 1 μs precision). They, they are transmitted through the USB to the PC in packets, which contain variable numbers of events, and when they are received, jAER applies a set of filters chosen by the user. Meanwhile, the software can render the events that are output by the final filter and add visual annotations over the output.

In this work, jAER was used to log AER-data[4] files from a Neuromorphic Auditory Sensor (Jiménez-Fernández et al., 2017) for further processing. Fig. 1.5 shows a screenshot from AER, receiving spikes in real time from the live output of a Neuromorphic Auditory Sensor and from a Dynamic Vision Sensor (DVS) (Serrano-Gotarredona and Linares-Barranco, 2013).

In this thesis, a new software tool for post-processing the output spiking information from neuromorphic auditory sensors was developed, and it is thoroughly explained in Appendix A.

[3] With "event-based data" we mean address-events from systems using AER protocol which have been timestamped.

[4] AER-data files have .aedat extension.

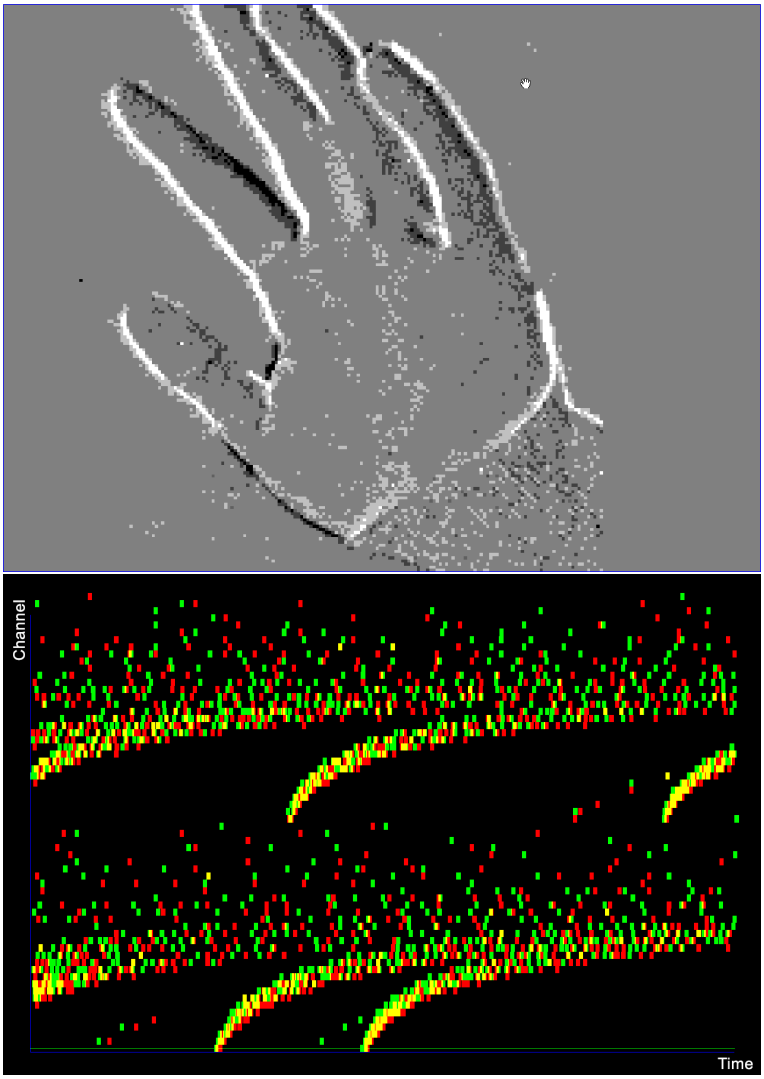


FIGURE 1.5: DVS output representation of a hand in motion in jAER (top) and NAS output representation of a speaker talking in jAER (bottom).

1.2.3.2 PyNN

PyNN (Davison et al., 2009) is a simulator-independent language for building neural network models using Python programming language. PyNN lets users build a network model by writing the code using its set of tools and functions and then run it without applying any modification to it on any of the simulators that

PyNN supports and on a number of neuromorphic hardware systems. Currently, PyNN supports NEURON (Hines and Carnevale, 2001), NEST (Kunkel et al., 2017) and Brian (Goodman and Brette, 2008) as simulators, and SpiNNaker (Furber et al., 2013) and BrainScaleS (Furber, 2016) as neuromorphic hardware systems.

The PyNN API provides a high-level of abstraction approach when modelling neural networks. The user is able to code a neural network model by only defining populations of neurons, and connections between them, using predefined cell models and commonly-used connectivity algorithms (all-to-all, random, one-to-one, among others) that the user does not need to implement. On the other hand, the user can also access lower level features like details of individual neurons and synapses when required. PyNN provides a library of standard cell models and synapse and synaptic plasticity models, along with the connectivity algorithms that were presented before, while still allowing users to build their own connectivity by coding it in Python.

1.2.3.3 NEST

As was previously introduced, NEST (Gewaltig and Diesmann, 2007; Kunkel et al., 2017) is an open-source (under GNU General Public License) spiking neural networks simulator. It is fast, memory efficient and has minimal dependencies. It is supported by PyNN as user interface, as well as by PyNEST. In this work it was used to model, train and test Spiking Neural Networks for audio samples classification before deploying them in a neuromorphic hardware platform. The neuromorphic hardware system used was SpiNNaker, which has a package called sPyNNaker that provides common code for PyNN implementations for SpiNNaker. Due to the fact that both NEST and SpiNNaker are supported by PyNN, modelling, training and testing a network in NEST and, after that, deploying the model in SpiNNaker, can be done with very few changes in the Python code. A more detailed description about SpiNNaker and its features is provided in section 1.2.5.

1.3 Auditory system and audio processing

In previous chapters, neuro-inspired systems were introduced. These systems mimic the way in which the senses and the brain process the information. Thus, it is intended to obtain benefits in the processing that are present in living organisms and that could be used for particular tasks like speech recognition and sound source localization. To be able to develop this kind of systems, it is necessary to have basic biological knowledge of the processing to be emulated. Therefore, in this chapter, the main characteristics of the sense of hearing

are presented, along with a state-of-the-art analysis of different bio-inspired neuromorphic auditory systems that have already been published, emphasizing the one that was used and that is the base of each of the classifiers that were developed in this work.

In this chapter, the elements of the auditory system that receive, analyze and encode the acoustic information into nerve impulses to be processed by the brain afterwards are presented first, including the mechanisms that make this whole process possible. Then, the key concepts of the main characteristics of a sound (pitch, timbre and loudness) are introduced, along with the three main processes of the analog-to-digital conversion: sampling, quantization and encoding. To conclude, a study about the computational models that represent the propagation of the sound through the inner ear and the conversion of the acoustic vibrations into nerve impulses is presented, along with some implementations of these models consisting of a study about the most relevant neuromorphic artificial cochleae.

Thanks to the study of the information presented in this chapter, especially on how the bio-inspired cochlea that was used works and processes the input analog sound signal, propagating this information using the AER protocol, a set of sound recognition systems and classifiers were developed, along with a software tool to post-process the raw spike information. Chapter 3 and the set of papers attached as appendices in this work detail each of the contributions that have been made.

1.3.1 Hearing in biology

In this chapter, the anatomy and the physiology of the auditory system are described, emphasizing the parts and structures that are more relevant for audio processing and sound recognition.

The concept of perception consists in the detection of a stimulus by one or more sensory receptors. The sensory receptors of the human body are continuously sensing things that we are not even aware of (e.g. blood pressure, blood temperature, carbon dioxide and oxygen concentration, etc.) due to the fact that the signals that are sensed are not sent to the region in the brain cortex that is in charge of consciousness. In other circumstances, perception leads to sensation (or conscious perception). In these cases, the sensory receptors transmit impulses to the brain cortex through nerves and pathways. The brain cortex is able to integrate these signals into a sensation in the order of microseconds (Hull, 2011).

Hearing is one of the senses that are classified as special senses, which have specialized organs devoted to them (vision has the eyes, hearing has the ears, smell has the nose and taste has the tongue). Therefore, hearing could be

defined as the detection of sound waves and their integration in order to generate sensations (Hull, 2011).

Every sensation, including the ones produced by hearing, are the result of the same sequence of events: first, a stimulus is produced. Then, the sensory receptor detects the stimulus and converts it into an electrical signal. Next, the signal is transmitted to the brain (through the auditory pathways in the case of hearing). Finally, the brain integrates the signal into conscious perception (sensation).

The task of receiving the audio signal, processing it and transforming it into impulses are carried out inside the ear, while neural processing and sound recognition are performed in the brain. Thus, two main regions or sections can be distinguished in the auditory system: the peripheral auditory system, where sound waves are converted into nerve impulses, and the central auditory nervous system, which transforms these impulses into sensations.

Different cognitive processes intervene in the central auditory nervous system, by which context and meaning are given to the sound, i.e., they allow the recognition of words, the classification of different musical instruments, the localization of the sound source, distinguishing between different speakers by their pitch and loudness, etc., among other complex tasks.

1.3.1.1 Peripheral auditory system

The peripheral auditory system, also known as the ear, is responsible for the physiological processes of hearing. These processes allow the reception of sound, transforming it into electrical impulses that are then sent to the brain through the auditory nerves. The mechanical processing of the sound waves and their transformation into impulses are non-linear processes (Zwicker and Fastl, 2013), which hinders the characterization and modelling of the auditory perception.

The peripheral auditory system is divided into three interconnected parts: the outer ear, the middle ear and the inner ear. Next, the anatomy and functionalities of each of these three parts of the ear are described, along with the propagations and the processing of the sound across them.

1.3.1.1.1 Outer ear

The outer ear is the external part of the ear and it consists of the auricle and the ear canal. The auricle is the visible part of the ear that resides outside the head (the word “ear” is also used to refer to this part alone) and it is also known as pinna. Its function is to gather the sound waves and guide them through the ear canal. When they hit the auricle, sound waves are reflected and attenuated, which provides additional information to the brain for determining

the direction of the sound that the auricle is receiving. The determination of the direction of the sound depends on the outer ear anatomy. Except for specific cases in which sounds come directly from towards, rearwards, above or below, the sound reaches the closest ear a fraction of a second earlier and louder than in the furthest one. If the difference is higher than 10 μs (minimum human interaural time difference threshold), the brain is able to detect and interpret it.

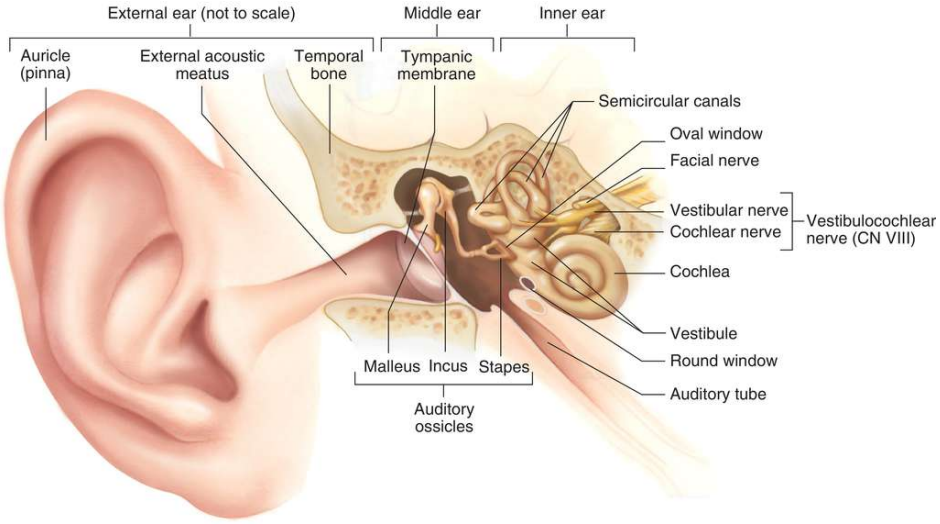


FIGURE 1.6: Anatomy of the ear. Image taken from (Patton et al., 2012).

The ear canal is 2.5 cm long and it extends from the auricle to the eardrum, also called the tympanic membrane, in the middle ear. Its main function is to protect the middle ear and to maintain the middle ear at a stable temperature. It conducts the vibrations gathered in the auricle to the tympanic cavity, amplifying sounds with frequencies between 3 and 12 KHz. Each of these components are shown in Fig. 1.6.

1.3.1.1.2 Middle ear

The middle ear is the part of the ear that is internal to the tympanic membrane, which separates this part from the outer ear (Fig. 1.6). The eardrum vibrates when it is hit by sound waves. The middle ear contains three small ossicles (malleus, incus, and stapes), which propagate the sound waves from the eardrum to the inner ear. The stapes is in contact with one of the fluids contained in the inner ear through the oval window. Thus, the ossicular chain acts as a mechanism to transform air vibrations into waves in the fluid and membranes of the inner ear. In order to achieve this, the air pressure inside

the middle ear must be the same as the atmospheric pressure, which is possible thanks to the Eustachian tube (also known as auditory tube or pharyngotympanic tube), connecting the middle ear with the nasopharynx and allowing pressure to equalize between the middle ear and throat (Hull, 2011; Patton et al., 2012).

The main functions of the middle ear are:

- To increase the pressure received by the eardrum. This is very important, since the cochlea is full of liquid, instead of air, and the density and compressibility of the cochlear liquid (the perilymph) is almost four thousand times lower than that of the air. If we did not have a mechanism to increase the pressure inside the middle ear, only 0.1% of the tympanic pressure would reach the cochlea. This acts as a sound normalization step phase. To protect the inner ear structures from extremely loud sounds, the stapedius (the muscle that stabilizes the stapes), stiffens the ossicular chain by pulling the stapes away from the oval window of the cochlea, decreasing the transmission of vibrational energy to the cochlea for sounds below 1-2 kHz and above 85-90 dB. This mechanism is known as stapedius reflex or acoustic reflex.
- To reduce the transmission of low-frequency sounds, acting as a low-pass filter, with an attenuation of 15 dB per octave in the 1 kHz band.

1.3.1.1.3 Inner ear

The inner ear is the innermost part of the peripheral auditory system. It consists of the vestibular system, which is dedicated to control balance, and the cochlea, dedicated to hearing.

The cochlea, which is shown in Fig. 1.7, is a 32-35 mm long, 4 mm^2 to 1 mm^2 wide (from the base to the apex) spiral-shaped cavity filled with two different fluids. Fig. 1.8 shows a cross section of the cochlea in which three tubular ducts can be seen. The central one is the cochlear duct (also known as scala media) and contains endolymph. The second and third ducts, called vestibular duct and tympanic duct (or scala vestibuli and scala tympani, respectively), contain the same fluid, perilymph, due to the fact that they are interconnected through a small opening in the apex of the cochlea called helicotrema. The base of the stapes is in contact with the fluid of the vestibular duct through the oval window, while the tympanic duct terminates at the round window in the tympanic cavity, as shown in Fig. 1.6. The lining between the cochlear duct and the vestibular duct is known as the Reissner's membrane, while the lining between the cochlear duct and the tympanic duct is called basilar membrane (Fig. 1.8) (Hull, 2011).

The basilar membrane is a structure whose width and rigidity are not constant: it is broad and rigid near the oval window, and it gets thinner and more flexible near the apex of the cochlea. Rigidity decays almost exponentially

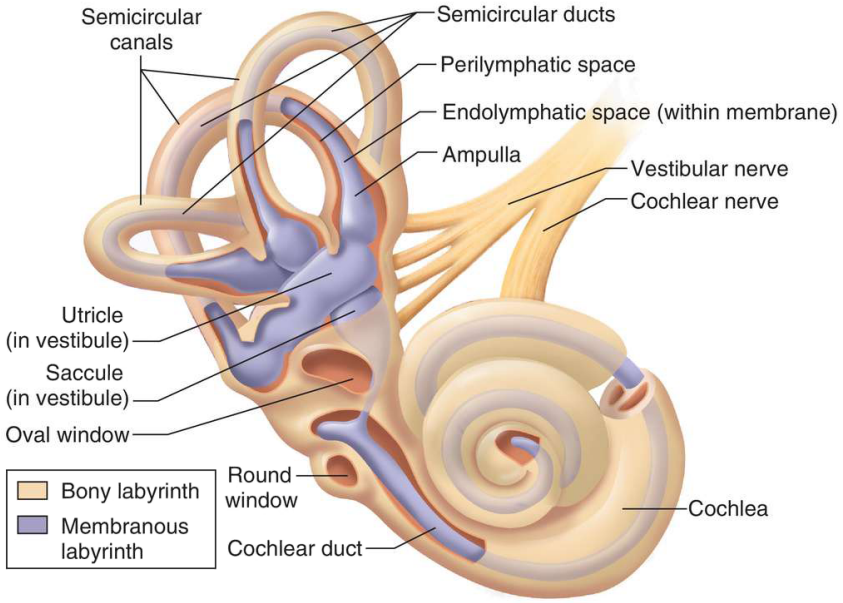


FIGURE 1.7: Inner ear. Image taken from (Patton et al., 2012).

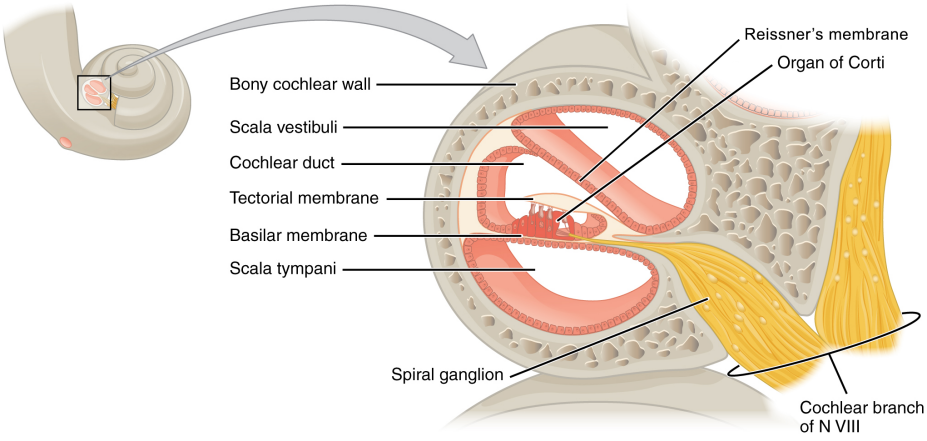


FIGURE 1.8: Cross section of the cochlea.

with the distance from the oval window. This variation affects the propagation speed of the sound waves across the basilar membrane, and it is responsible for one of the most important functionalities of the inner ear: frequency selectivity.

The basilar membrane supports the Organ of Corti (also known as spiral organ), the most important element in the cochlea, which is able to transduct from

movement into nerve impulses' action potential (see Fig. 1.9). The Organ of Corti contains between 15000 and 30000 receptors called hair cells. Their name derives from the tufts of stereocilia, known as hair bundles, which protrude from the apical surface of the cell into the cochlear duct. Each of the "hairs" of these cells are able to produce receptor potentials when they touch the tectorial membrane, which is located above the basilar membrane.

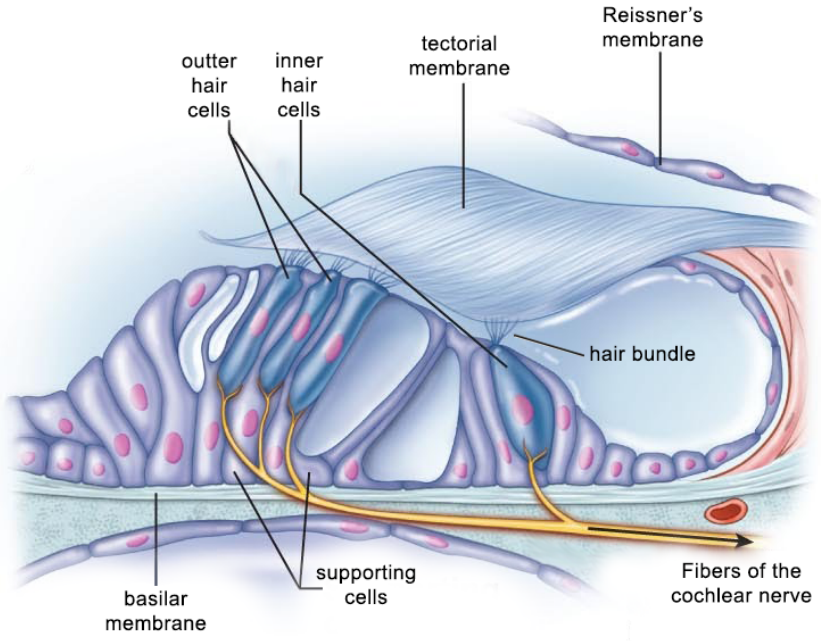


FIGURE 1.9: Inner structure of the Organ of Corti.

Hair cells can be divided into two different types: the inner hair cells (IHCs) and the outer hair cells (OHCs). There exist around 3500 IHCs and 20000 OHCs. Both cell types have connections with the afferent nerve fibers, which propagate action potentials towards the brain, and efferent nerve fibers, which propagate action potentials from the brain to the cochlea. However, the fiber distribution is very unequal: more than 90% of the afferent fibers innervate IHCs, whereas most of the 500 efferent fibers innervate OHCs. The functionality of each type is presented next.

The behavior of the cochlea starts with the vibrations produced by the stapes, which generates vibrations in the fluid contained in the vestibular duct. Oscillations in the perilymph of the vestibular duct are transmitted to the endolymph, and then to the basilar membrane, which also propagates the oscillations to the fluid in the tympanic duct (see Fig. 1.10). It is important to

note that the amplitude and frequency of the vibrations are directly proportional to the amplitude and frequency of the sound waves.

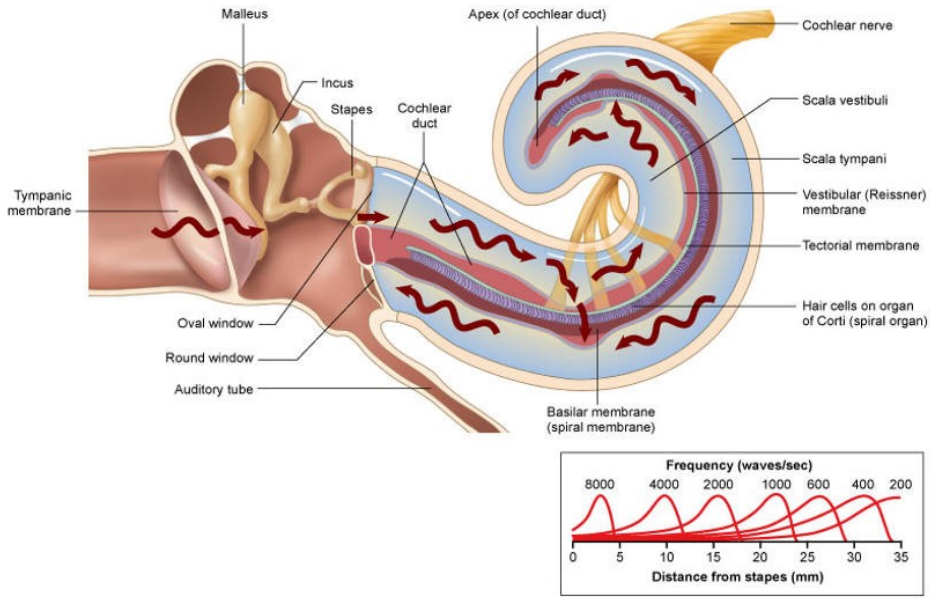


FIGURE 1.10: Effect of sound waves on cochlear structures. Image taken from (Patton et al., 2012).

The sound wave that generates these oscillations has a peak value in its amplitude in a specific region of the cochlea that depends on the frequency of the wave and it tends to decrease rapidly near the apex. The lower the frequency of the sound is, the greater the distance that the wave will travel across the membrane before being attenuated, and vice versa. This way, the basilar membrane scatters the different frequency components of a complex spectrum signal in well-defined positions with respect to the oval window, as is shown in Fig. 1.11.

The waves that are propagated through the cochlea produce a force or pressure to the cochlear duct, and therefore to the Organ of Corti. In the Organ of Corti, the hair cells rest upon the basilar membrane, and the external part of the stereocilia are in contact with the tectorial membrane (see Fig. 1.9). Both membranes have different flexibility, making each of them move in relation to the other one when a wave travels across them. As a result, the external part of the hair cells bend when the membranes move. This bending produces a change in the action potential of the hair cells: depending on the bending direction, the hair cell hyperpolarizes or depolarizes. These variations in the action potential produce changes in the neurotransmitter release of the hair cells in the synapse

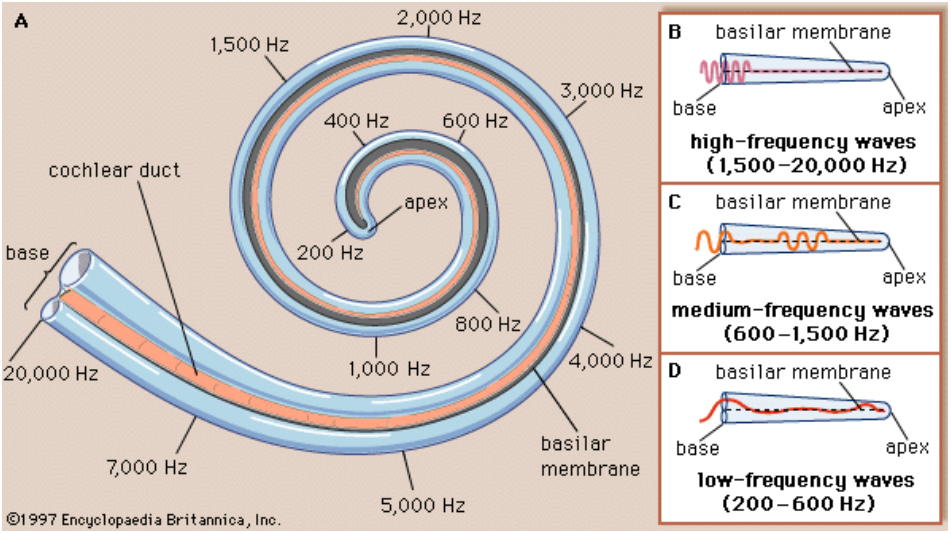


FIGURE 1.11: Tonotopic distribution of the cochlea (A). Localization of high-frequency (B), medium-frequency (C) and low-frequency (D) responses in the cochlea.

with a first order neuron. The molecules of the neurotransmitters released in the synapse change the action potential of a neuron of the vestibulocochlear nerve, altering the impulse frequency of the action potentials. These action potentials travel to the brain through the vestibulocochlear nerve (Hull, 2011).

The intensity of the auditory stimulation depends on the number of action potentials per unit of time and the number of cells that have been stimulated, while the frequency of the signal depends on which specific populations of nerve fibers are activated. There exists an association between the input sound frequency and the section of the cerebral cortex that has been stimulated. The lower the frequency of the vibration of the sound is, the closer to the apex the maximum excitation of the basilar membrane will take place. For greater frequencies, the maximum excitation will take place near the oval window. Depending on the section of the basilar membrane that oscillates with higher amplitude, the hair cells of that section will be activated in a higher proportion, stimulating subsequent afferent neurons that will produce spikes. This process originated the concept of characteristic frequency, to describe the way in which neurons in the middle ear respond with a particular low threshold for sound waves with a specific frequency, and plays an important role in the tone discrimination of a sound. If the sound wave in the input corresponds to a pure tone, a specific region of the basilar membrane with a particular characteristic frequency will oscillate with a higher amplitude. On the other hand, the further the section of the basilar membrane is from the characteristic frequency of the

pure tone, the weaker the response will be. Thus, each of the sections of the basilar membrane act as an auditory filter that reacts to a narrow frequency bandwidth (critical band).

A higher amplitude sound will produce a higher amplitude wave in the basilar membrane, increasing the number of IHCs that are excited, along with the number of action potentials that are generated in the afferent neurons. The difference between the IHCs and the OHCs lies on their functionality. While IHCs are in charge of transforming the amplitude of the wave into action potentials (as was explained in previous lines), the main functionality of OHCs is the ability to adapt the cochlear response depending on the input stimulus received. Thus, because of OHCs, the cochlear response is non-linear (see Fig. 1.12). The cerebral cortex classifies tones based on the region of the cochlea that has been excited, and their amplitudes based on the number of active neurons and their firing rate.

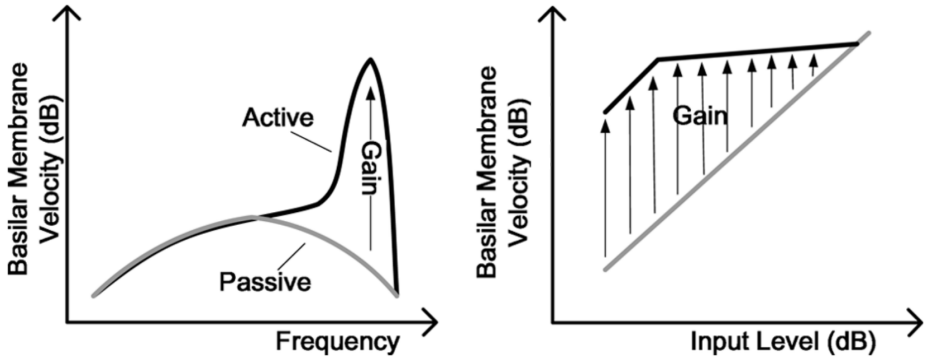


FIGURE 1.12: Effects of the nonlinear behavior of the cochlea on Basilar Membrane Velocity. (left) Response of a point on the basilar membrane without the effect of OHCs (Passive) and with OHCs (Active). (right) Response level as a function of input level at the characteristic frequency of the basilar membrane section.

Summarizing, two types of signal representation can be found in the auditory nerve: the spectral representation and the temporal representation. This duality is caused by the fact that the hair cells of the cochlea, which present a tonotopic organization, generate a different response based on the amplitude of the signal and its temporal envelope. Therefore, for a pure tone with a specific frequency, it is represented in the auditory nerve by a position, based on the position of the IHCs that are excited with that frequency, and by the periodicity of the responses of the fibers that react to that stimulus (temporal representation).

1.3.1.2 Central auditory nervous system

The central auditory nervous system consists of the auditory pathways and the auditory cortex. The signals generated in the Organ of Corti inside the cochlea are sent through the vestibulocochlear nerve to the auditory cortex. The auditory cortex is the part of the temporal lobe of the brain that processes auditory information in humans and other vertebrates. It represents the highest level of the auditory system in mammals.

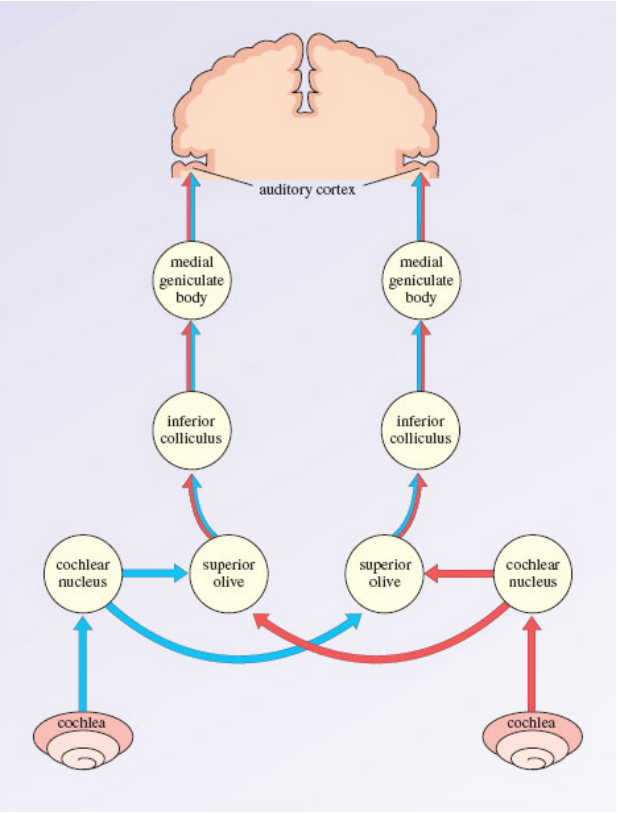


FIGURE 1.13: Highly schematic diagram of the bilateral central auditory pathway. The main pathways and nuclei are shown for both cochleae. Binaural stimulation occurs at the superior olive and all regions above.

From the vestibulocochlear nerve to the auditory cortex, the signal travels through the ascending auditory pathway (also known as the afferent pathway), where other important tracts and nuclei of the central auditory nervous system exist. Fig. 1.13 shows a highly schematic and simplified diagram with only the main components, although other nuclei exist. Almost all fibers of the auditory nerve synapse on cells of the cochlear nucleus (comprising the ventral

cochlear nucleus and the dorsal cochlear nucleus), where the processing of the acoustic information begins. Signals can take different paths in their way from the cochlear nucleus to the auditory cortex. Most of the axons of the cochlear nucleus cells (around 70%) cross over to the opposite side (contralateral side) of the brain, whereas only 30% of them are connected with other elements in the same side of the brain as the ear from which the signal was received. Thus, each of the hemispheres of the brain cortex receives auditory information from both ears (Hull, 2011).

Both crossed and uncrossed axons of the cochlear nuclei synapse in an area of the central auditory nervous system called the superior olivary complex, which is the first place in the afferent pathway of the auditory system that receives information from both ears. The superior olivary complex is divided into three main different nuclei: the medial superior olive, the lateral superior olive and the medial nucleus of the trapezoid body. The medial superior olive is believed to measure the time difference of the arrival of sounds between the ears (interaural time difference or ITD), whereas the lateral superior olive is believed to be involved in measuring the difference in sound intensity between both ears (interaural level difference or ILD). Both ITD and ILD are very important for determining the azimuth of sounds, i.e. localizing the input sound.

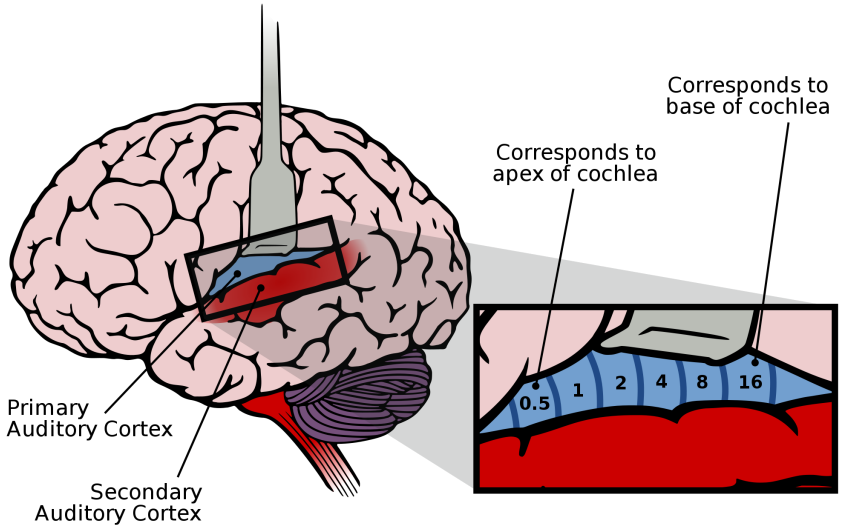


FIGURE 1.14: Lateral view of the human brain, with the auditory cortex exposed. The primary auditory cortex contains a topographic map of the cochlear frequency spectrum (shown in kilohertz). Author of the image: Chittka L, Brockmann.

Then, the impulses are transmitted to the inferior colliculus, whose main functionalities are signal integration from various auditory nuclei, frequency

recognition and pitch discrimination (Skottun et al., 2001; Shore, 2009). From there, the information goes to the medial geniculate body (also known as medial geniculate nucleus), which is part of the auditory thalamus and serves as a connection between the inferior colliculus and the auditory cortex. Neurons in the auditory cortex are organized based on the frequencies of the sound to which they respond best, e.g. a frequency map (known as tonotopic map), as shown in Fig. 1.14. The impulses generated by the hair cells in a particular section of the basilar membrane (sensitive to a specific frequency interval) are projected in the corresponding region in the auditory cortex. Therefore, the brain perceives the tone received in the ear based on the area of the cortex that is stimulated. The auditory cortex also has neurons that respond best to the action potentials generated by high amplitude sounds, while others are more sensitive to low amplitude sounds. Part of the auditory cortex is responsible for giving sense to the information, discriminating between different sound patterns based on its variations in tone, frequency, intensity and direction. The auditory cortex also receives information from other regions of the brain such as the visual cortex and the somatosensory cortex, collaborating in the interpretation of the signals (Hull, 2011).

The auditory system also transmits information from the auditory cortex to the cochlea through the efferent pathway (also known as the descending auditory pathway), giving feedback and modifying the analysis that is made in the cochlea in the form of frequency discrimination or non-linear amplification of quiet sounds. OHCs play an important role in this process (Cant and Benson, 2003).

1.3.1.3 Psychoacoustics

Acoustics is the branch of physics that describes the physical characteristics of the sound, while psychoacoustics studies how the sounds are perceived. The main goals of psychoacoustics is the study of hearing through the subjective responses to auditory stimuli.

Hearing is not a purely mechanical phenomenon of wave propagation; it is also a sensory and perceptual event.

In the same way as the intensity of a sound and the loudness (the intensity with which we perceive the sound) differ, the frequency of the sound and the frequency with which we perceive that sound can also be differentiated. The frequency components of a sound is objective information; however, the subjective interpretation of the frequency of a sound is called pitch. Pitch has been defined by ANSI (American National Standards Institute) as "that auditory attribute of sound according to which sounds can be ordered on a scale from low to high". For complex sounds like the ones produced by musical notes,

which consist of a fundamental frequency and one or more harmonics, the pitch is defined only by the fundamental frequency.

This work presents a system that classifies between different pure tones. In this case, frequency and pitch will be considered synonyms.

Timbre is defined by ANSI as "that attribute of auditory sensation in terms of which a listener can judge that two sounds similarly presented and having the same loudness and pitch are dissimilar", and by Risset and Mathews (Mathews et al., 1969) as "the attribute that enables the listener to identify the instrument producing the tone". Thus, timbre is the subjective quality that allows us to distinguish between different musical instruments that produce the same musical note. Two musical notes with the same fundamental frequency could have a different spectrum because of their harmonics.

Humans are able to perceive sounds in a frequency region between 20 Hz and 20000 Hz, although we are more sensitive to the ones between 2000 Hz and 5000 Hz. This range differs between two different persons, shrinking during life, particularly the perception of high frequency sounds (Rodríguez Valiente et al., 2014).

The frequency resolution of the ear depends on the intensity and the frequency of the sounds. For tones around 200 Hz we are able to discriminate sounds with minimum frequency differences of 1 Hz. On the other hand, for higher frequencies, the frequency resolution increases. For example, for a 10 kHz tone, the frequency resolution is 200 Hz. The reason for this is that the auditory system acts as a set of overlapped filters, in which lower frequencies involve narrower filters and higher frequencies involve wider filters (Janus, 2004).

In silent environments, humans can hear sounds with an intensity range between 0 dB-SPL (Sound Pressure Level) and 130 dB-SPL. Sounds with an intensity greater than 130 dB-SPL could damage the ear. The minimum intensity differential threshold between two sounds is called JND (Just-Noticeable Difference) and has a value of 1 dB (Janus, 2004). Fig. 1.15 shows the perceived human hearing.

Regarding the duration of the sound, it only has an inferior limit. The shortest perceptible sound ranges between 10 and 40 ms (Bascuas, 1997). Twenty milliseconds is a widely used value when integrating information of sounds in auditory processing.

To represent loudness in a graphical way against the frequency and intensity of the tones, equal-loudness contours are used. This metric was developed by Fletcher and Munson (Fletcher and Munson, 1933). These curves calculate the existing relation between the frequency and intensity (dB) of two sounds, in order for these to be perceived equally loud by the ear, meaning that each dot of the same contour have the same loudness (Janus, 2004).

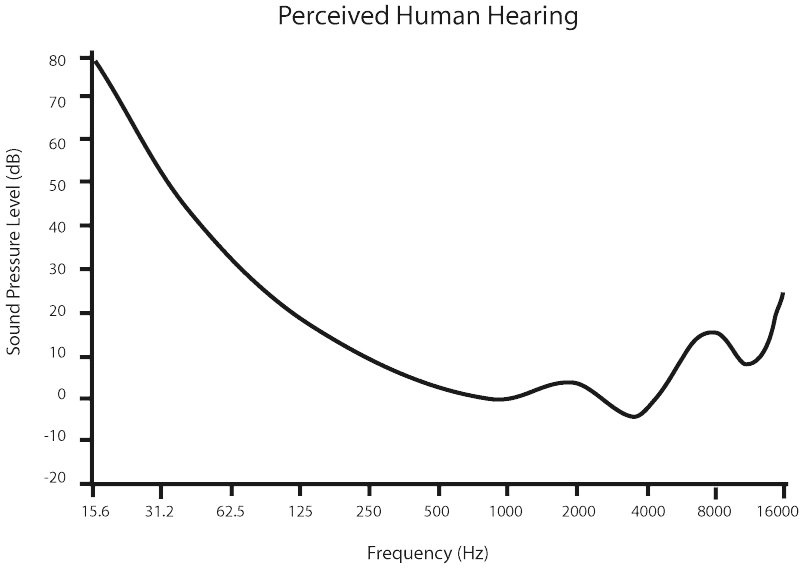


FIGURE 1.15: The absolute threshold of hearing. The curve shows the quietest sounds a human can hear depending on the frequency and the intensity of the sound.

There is also a relation between the duration and loudness of a sound. A sound with a constant intensity decreases its loudness by 25% after some minutes (Janus, 2004).

One noticeable characteristic of the auditory system is the masking effect (O'shaughnessy, 1987). Auditory masking is the effect by which the perception of one sound is affected by the presence of another sound. Auditory masking in the frequency domain is known as simultaneous masking, frequency masking or spectral masking. Auditory masking in the time domain is known as temporal masking or non-simultaneous masking.

Frequency masking experiments led Fletcher on the hypothesis that part of the auditory system behaves as a set of overlapped band pass filters (Fletcher, 1940). Each of these filters corresponds to a section of the basilar membrane. Each section is 0.9 mm long approximately. The range of these filters that stimulate the same portion of the basilar membrane is called critical band. While some studies have described a specific number of critical bands with well-defined ranges, like the Bark scale (Zwicker, 1961), Mel scale (Stevens et al., 1937) and ERB scale (Equivalent Rectangular Bandwidth) (Glasberg and Moore, 1990), other studies have suggested that the cochlea is a set of continuous filters. The next section describes different auditory system models and how they are implemented in

analog and digital electronic systems, where critical bands take an important role.

1.3.2 Bio-inspired electronic auditory systems

Artificial cochleae model the basilar membrane using a set of filters or resonators with cutoff/center frequencies (depending on whether low-pass filters or band-pass filters are implemented), which mimic the frequency distribution along the basilar membrane. Depending of the position of the basilar membrane, its flexibility and width changes. These changes are implemented by setting different parameters to the filters.

As has been presented in previous sections, the biological cochlea is a very complex part of the inner ear, which means that modelling and implementing its functionality is not an easy task. This fact has resulted on artificial cochleae implementations that only model part of the characteristics of the biological cochlea. These characteristics are chosen depending on the application that researchers want to integrate in the system. Therefore, the majority of artificial cochleae do not achieve comparable results to the ones obtained by a biological cochlea.

Since the first design of an artificial cochlea presented by Richard Lyon and Carver Mead in 1988 (Lyon and Mead, 1988), activities referred to the implementation of different mathematical models of the cochlea using analog VLSI have increased. Digital implementations using reconfigurable logic devices have also increased in popularity, although not as much as analog approaches.

Even three decades after the first silicon cochlea, artificial cochleae are still far from being comparable to the biological cochlea, especially in terms of power consumption, frequency range, dynamic range of the input or noise immunity. Taking into account that these models aim to emulate in the same way as a system that has evolved for hundreds of millions of years, some good approximations have been achieved.

Fig. 1.16 shows different implementations of artificial cochleae and their evolution in time.

Artificial cochleae can be categorized in many different ways. Generally, they are classified based on:

- The coupling coefficient that exists between the elements of the filters. We distinguish between one-dimensional silicon cochlea (1-D) and two-dimensional silicon cochlea (2-D).
- The existence or lack of automatic gain control to allow filters to dynamically adapt to the intensity changes in the input (active or passive cochleae).

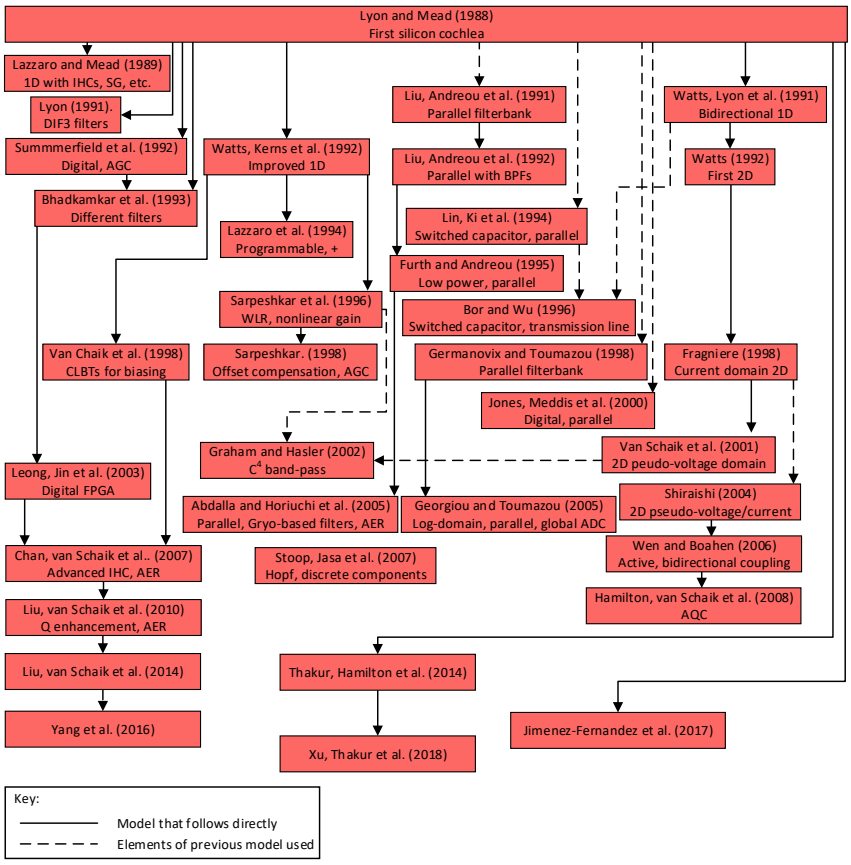


FIGURE 1.16: Historical tree diagram of different artificial cochleae developed.

In this section the most relevant developments regarding artificial cochleae are presented, both analog and digital implementations. Each of them is based on mathematical models that represent the propagation of the sound wave through the inner ear and the conversion between acoustic energy and nerve impulses, for later processing. Some of the most popular mathematical models of the auditory system are explained in this section before describing analog and digital cochleae implementations.

1.3.2.1 Auditory system models

As has been detailed in previous sections, part of the human auditory system acts as a continuous and overlapped set of band pass filters, which correspond

to a specific region of the basilar membrane. In this section, the most relevant models that represent this behavior of the cochlea are presented. In addition to the behavior of the cochlea, some of these models also include other parts of the peripheral auditory system.

Based on the structure of the band pass filters that are implemented, two different auditory models can be found: parallel (independent filters that process the input information at the same time) or cascade (set of filters that are connected, where the input of one of them is the output from the previous one in the cascade).

In the response of the critical bands of the basilar membrane, a very pronounced peak can be observed, which corresponds to the resonant frequency, and an attenuation for frequencies above or below the resonant frequency, with the slope being steeper for higher frequencies. This effect can be achieved with a cascade band pass filter. A parallel band pass filter would need higher order filters to implement that behavior.

1.3.2.1.1 ERB model

The ERB (Equivalent Rectangular Bandwidth) model (also known as Patterson-Holdsworth model) (Slaney, 1993) is based on the research by Roy D. Patterson and John Holdsworth about the cochlea and the inner ear (Patterson et al., 1992). It consists of a set of band pass filters in a parallel or independent configuration. Each of them is tuned with a different frequency. In this model, the bandwidth of each auditory filter is set with an equivalent rectangular bandwidth. The main idea of this approach is to approximate the critical bands of the cochlea using band pass filters with an equivalent rectangular bandwidth whose height is the response in magnitude of the filter and whose area is the same as the response of the filter (see Fig. 1.17). A critical bands filter, or ERB, models the signal that is present in only one cell of the auditory nerve. The ERB shows the relation between the auditory filter, frequency, and the critical bandwidth. An ERB passes the same amount of energy as the auditory filter it corresponds to and shows how it changes with input frequency (Gelfand, 2017).

The bandwidth of a critical band was first approximated by Glasberg and Moore (Glasberg and Moore, 1990) using ERB. At low sound levels, this value can be approximated by the following expression (Equation 1.1):

$$ERB(f_c) = 24.7(4.37\frac{f_c}{1000} + 1) \tag{1.1}$$

Where both the ERB and f_c (the center frequency) are in Hz.

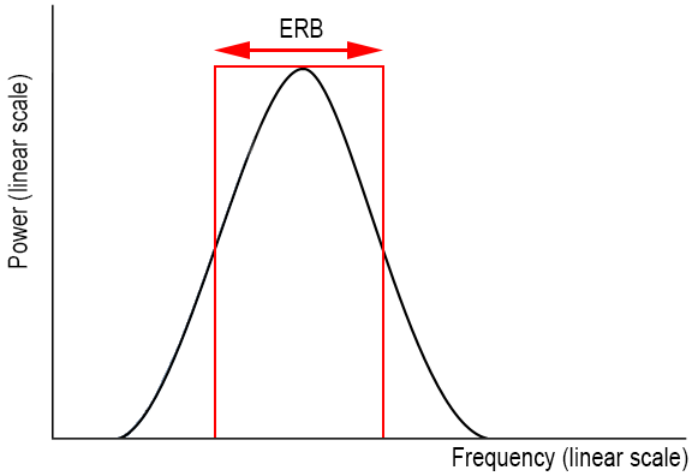


FIGURE 1.17: ERB representation.

Due to the fact that the architecture is purely parallel, there is no dependency between successive filters, as in other approaches based on a cascade architecture. Fig. 1.18 shows the frequency response of this model based on the implementation of Malcolm Slaney (Slaney, 1993).

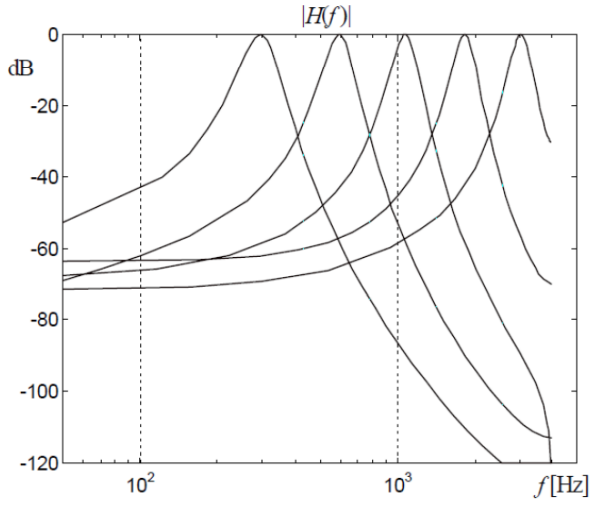


FIGURE 1.18: Frequency response of gammatone filters (order 8, N=4) for five characteristic frequencies: 3.03, 1.83, 1.07, 0.6, 0.3 kHz. Image taken from (Miró Amarante, 2013).

This model consists of gammatone filters with responses to the impulse. The significance of these filters for hearing is that they can generate a frequency response that is very similar to the one produced by the human auditory filters. They can represent how the basilar membranes react to a particular stimulus. The main downside of this model is that the responses are very symmetric, i.e., there are no differences between the slopes of the increasing and decreasing attenuation with respect to each resonant frequency.

In this model, the frequency separation between different channels is not specified. Automatic gain control is not implemented or taken into account in this model.

1.3.2.1.2 Lyon's model

A cochlear model based on the knowledge of the biological cochlea and its main functionalities was developed by Richard F. Lyon (Lyon, 1982). This model describes the propagation of sound in the inner ear and the conversion of acoustic energy into neural representations. When sound reaches the cochlea, a wave travels through the basilar membrane. The physical properties of the basilar membrane change from the base where the oval window is to the apex, so that the frequency components of the wave reach a maximum in a particular position of the basilar membrane.

The cochlear model described by Lyon combines a set of filters, to represent the waves that travel through the basilar membrane, half-wave rectifiers (HWR), to detect the energy of the signal, acting as IHCs, and different automatic gain control (AGC) stages, to model the behavior of the OHCs. Due to the fact that the fundamental frequency of the basilar membrane decays exponentially from the base to the apex, the basilar membrane is divided into sections with the same length to obtain the frequency distribution of the auditory filters.

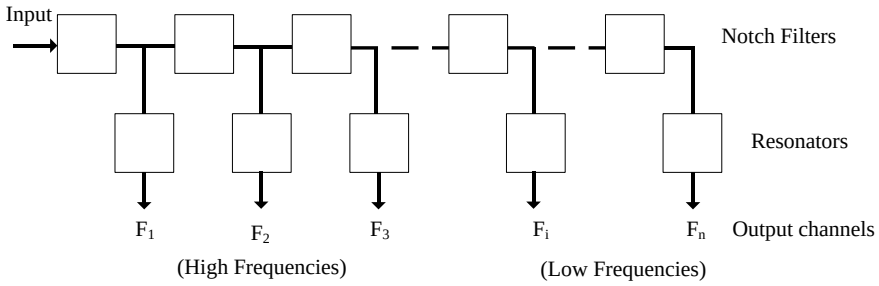


FIGURE 1.19: Block diagram of the filters in Lyon's model.

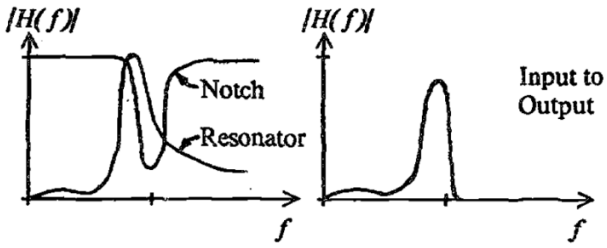


FIGURE 1.20: Transfer function of the filters used in the filterbank. Image taken from (Lyon, 1982).

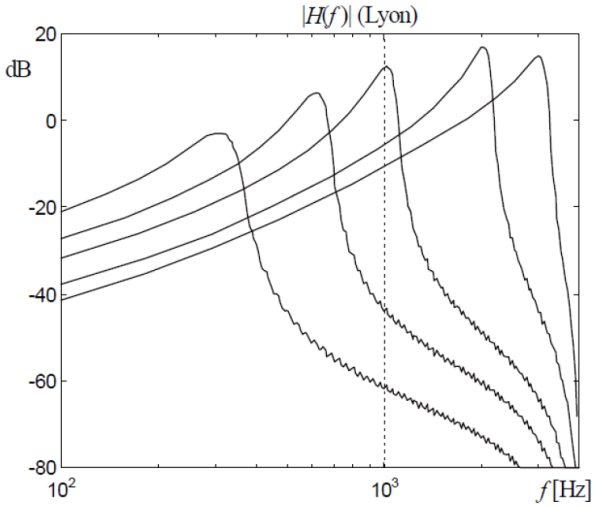


FIGURE 1.21: Frequency response of Lyon's model (64 sections) for the following characteristic frequencies: 3.0, 2.0, 1.0, 0.6 and 0.3 kHz. Image taken from (Miró Amarante, 2013).

At each point in the cochlea, the acoustic wave is filtered by a notch filter[5]. Each notch filter operates at successfully lower frequencies so the net effect is to gradually low-pass filter the acoustic energy. An additional resonator (or bandpass filter) picks out a small range of the traveling energy and models the conversion into basilar membrane motion. It is this motion of the basilar membrane that is detected by the inner hair cells. The block diagram that represents this cascade architecture is shown in Fig. 1.19. This way, the high frequency components of the signal are filtered while the remaining components of the signal travel through the cascade of filters. The rejection of the high

[5]The notch filter, also known as band-stop filter or band-rejection filter, is a filter that passes most frequencies unaltered, but attenuates those in a specific range to very low levels.

frequency components produces a steep slope in the frequency response of the filters that correspond to low frequencies. The biological cochlea also presents this behavior. Therefore, this model provides a good approximation to the processing that is done in the cochlea. Fig. 1.20 shows the transfer function of the notch filter, the resonator and the combination of both.

Fig. 1.21 presents the response of the Lyon's model for a configuration with 64 sections and a sampling frequency of 8 kHz. It is important to highlight the difference between the ascending and the descending slopes with respect to the fundamental frequency. The higher the fundamental frequency, the higher the slope immediately after. This effect is due to the rejection of the high frequency components across the cascade of filters. The attenuation of the signal at low frequencies is caused by the inclusion of a preemphasis filter, which roughly models the effects of the outer and middle ear.

1.3.2.1.3 Lyon & Katsiamis' model

Richard Lyon, together with Andreas Katsiamis and Emmanuel Drakakis, published a research paper in 2007, in which they presented transfer functions in the continuous domain that were designed based on gammatone filters for auditory information processing (Katsiamis et al., 2007).

In the paper, the design of two different type of filters is presented: the Differentiated All-Pole Gammatone Filter (DAPGF) and the One-Zero Gammatone Filter (OZGF). These designs are characterized for having a hardware-implementation oriented architecture; they have the same properties and functionalities as the cochlea operation and overcome some limitations of the gammatone filters, such as the symmetric response and the complexity in the frequency domain (Miró Amarante, 2013).

Unlike the cascade architecture of Lyon's model, this model is based on a bank of filters with parallel stages that consist of blocks connected in a cascade fashion. Fig. 1.22 shows how the basilar membrane can be modeled with both a parallel or a cascade architecture.

1.3.2.1.4 Inner hair cells model

To create a fully neuromorphic model of the cochlea, some elements have been added to mimic the behavior of the inner hair cells (IHC). As was presented and detailed in section 1.2.3.1.1.3., IHCs convert the vibrations of the basilar membrane into electrical signals (impulses).

There exist several IHC models, among which the most popular and most used is the one proposed by Meddis in 1986. This IHC computer simulation

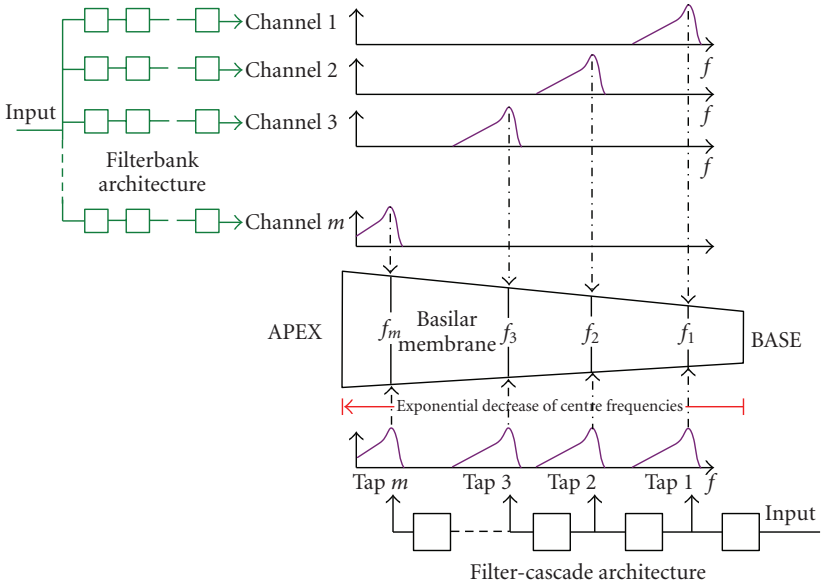


FIGURE 1.22: Graphical representation of the filterbank and filter-cascade architectures in the Lyon-Katsiamis model. Image taken from (Katsiamis et al., 2007).

model (Meddis, 1986; Meddis, 1988; Meddis et al., 1990) is widely accepted and it was the base for subsequent models implemented in aVLSI (advanced Very Large Scale of Integration) (McEwan and Schaik, 2003; McEwan and Schaik, 2004).

1.3.2.2 Analog implementations

A one-dimensional cascade cochlea models the propagation of the sound wave through the basilar membrane in one unique direction (from the base of the basilar membrane to the apex). Moreover, in this cascade architecture, each section of the cochlea (auditory filter) processes the output of the previous element. Even when using second-order filters, this allows a steep slope effect to take place, which favors the frequency selectivity of the basilar membrane itself.

The first bio-inspired artificial cochlea was developed by Lyon and Mead in 1988 (Lyon and Mead, 1988) following this one-dimensional cascade topology, the Lyon's model, which was presented in section 1.2.3.2.1.2. (Lyon, 1982). It has been proved that, in the human cochlea, the resonant frequency across the basilar membrane decreases exponentially in a logarithmic scale: high frequencies near the base and low frequencies near the apex. To implement this artificial cochlea, the basilar membrane has been divided into different segments with the same

length. A set of filters with a specific resonant frequency according to the resonant frequency of each segment of the basilar membrane is used in a cascade topology. Each of the filters is the same, only changing the resonant frequency and their low-pass response. High frequency components are removed in the output of each filter, which produces a steep slope in the frequency response curves. This slope is also observed in the biological cochlea when the wave travels across the basilar membrane. Therefore, despite the simplicity of this model, it provides a first approximation of signal processing inside a biological cochlea.

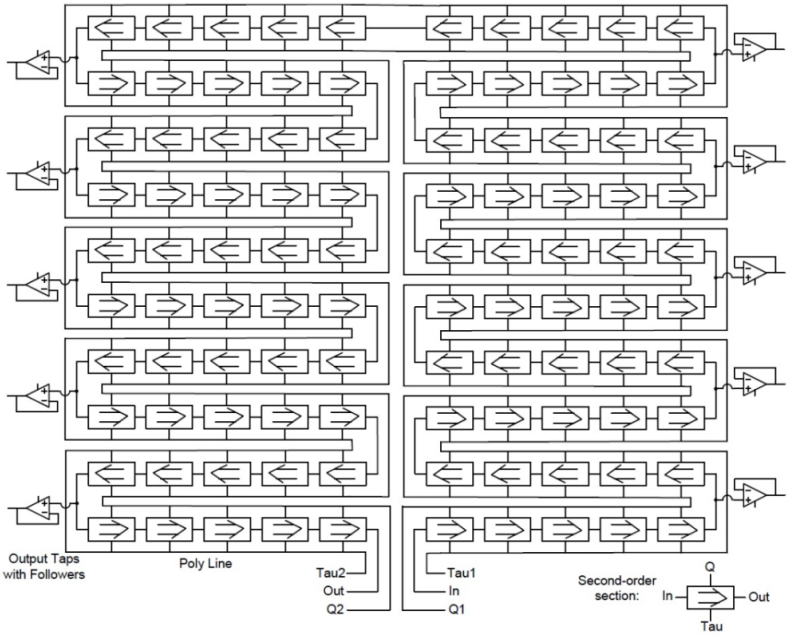


FIGURE 1.23: Floorplan of 100-stage Lyon and Mead's cochlea chip, in serpentine arrangement. Image taken from (Lyon and Mead, 1988).

The work developed by Lyon and Mead consists of 480 sections of second-order low-pass filters in a cascade topology with resonant frequencies logarithmically distributed in order to model the propagation of the wave and the frequency analysis associated to the basilar membrane. Fig. 1.23 shows a diagram of a 100 stage cochlea following the Lyon and Mead's cascade structure. This cochlea implementation successfully modeled particular characteristics of the biological cochlea, and has provided a starting point for the research of neuromorphic cochleae. There exist several implementations based on this first one proposed by Lyon and Mead.

John Lazzaro added circuits to the Lyon and Mead's cochlea in order to model the behavior of the IHCs. These circuits encode the output spikes of the

artificial cochlea (Lazzaro and Mead, 1989).

Another implementation based on this approach is the work by Lloyd Watts (Watts et al., 1992), which achieves a better exponential distribution of the resonant frequencies. This increments the linear range and eliminates the instability of the signal. Fig. 1.24 shows the frequency response of each of the stages of the cochlea before and after these circuits are included. In 1995, Lazzaro and Wawrzynek (Lazzaro and Wawrzynek, 1995) proposed an improved version of the Watts' model, adding the AER communication protocol in the output of the cochlea. It is the first artificial cochlea to use this kind of communication, which currently is widely extended in neuromorphic developments.

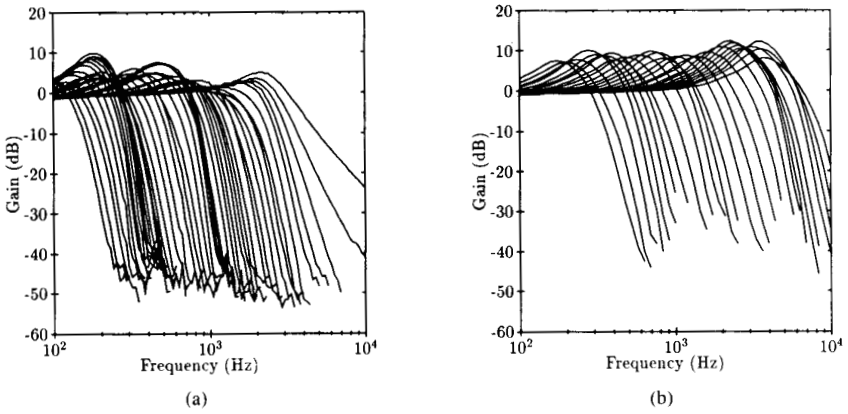


FIGURE 1.24: Frequency response of the filters in (a) Lyon's layout and (b) Watt' improved layout. Image taken from (Watts et al., 1992).

In 1996, Andre van Schalk, Eric Fragniere and Eric Vittoz presented a new artificial cochlea implementation based on Watts' (Van Schaik et al., 1996). Fig. 1.25 shows the cutoff frequencies distribution of this implementation compared to the cutoff frequencies distribution achieved in Watts' work. As can be observed, in this implementation the exponential distribution of the frequencies is more uniform. Thanks to this improvement, this artificial cochlea allows binaural sounds processing, as is presented in works like (Chan et al., 2007) and (Yu et al., 2009), in which two van Schaik's artificial cochleae with a more advanced module to manage the output spikes by using the AER protocol are used for echolocation tasks. The artificial cochlea that was presented in (Chan et al., 2007) has also been used to build a classification system in order to distinguish between two sounds: a clap and a bass drum (Jäckel et al., 2010). This cochlea was improved in a number of channels, output event rate and frequency range in (Liu et al., 2010) and (Liu et al., 2014).

The main downsides of the cascade topology is the low fault-tolerance. If any of the elements of the cascade fails, this error will be propagated to the rest

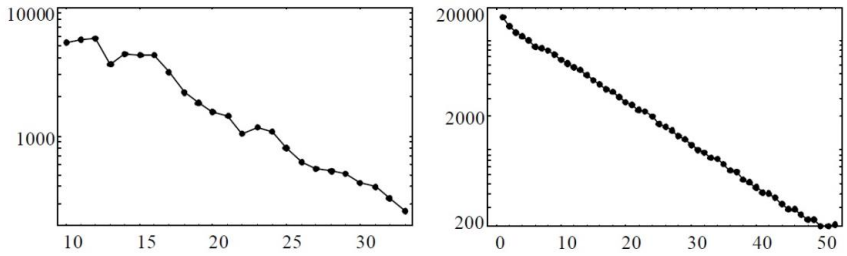


FIGURE 1.25: Cutoff frequencies (Hz) distribution in Watt's artificial cochlea (left) and van Schaik's (right). Image taken from (Van Schaik et al., 1996).

of the subsequent elements. Also, it is important to notice that each segment of the cascade adds a delay, which is inversely proportional to the center frequency of the filter. Then, filters corresponding to low frequencies will introduce more delay than the ones corresponding to high frequencies. Therefore, the number of sections (stages or channels) limits the frequency response of the system. The noise that is generated by the filters is accumulated across the cascade, which reduces the dynamic range of the system. Some of these issues can be solved with a parallel topology or a 2-D topology.

Parallel topologies are usually chosen for their ease of implementation. However, even though the issues of cascade topologies are not present in these models, parallel topologies are not the best option when implementing an analog cochlea because each filter acts independently from the rest and higher order filters are needed to produce the same "steep slope" effect in the high frequencies as in cascade topologies, which increments the power consumption of the system, requiring a larger silicon area. Fig. 1.26 presents a comparison between the outputs of a single second-order filter in a 1-D parallel topology (Fig. 1.26 (a)), and in a 1-D cascade topology (Fig. 1.26 (b)).

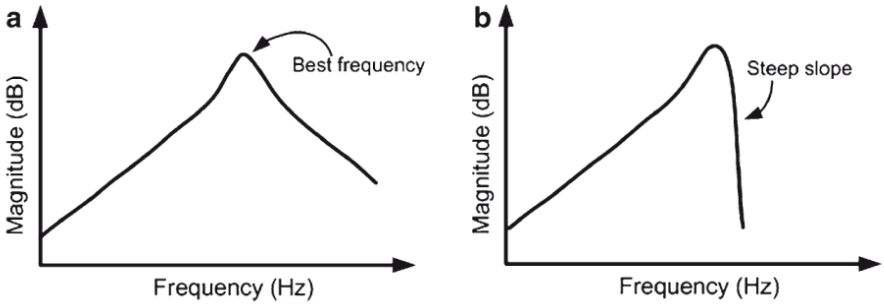


FIGURE 1.26: Frequency response of a second order filter in a parallel topology (a) and in a cascade topology (b).

The implementations presented in (Liu et al., 1991) and (Liu et al., 1992) are some examples of parallel topologies. Following the same architecture that was implemented in these works, in 1998 a new artificial cochlea was developed for pattern extraction in speech recognition tasks (Kumar et al., 1998). In 2016, Minhao Yang, Chen-Han Chien, Tobias Delbruck and Shih-Chii Liu presented a 64-channel binaural cochlea with parallel asynchronous event output (Yang et al., 2016). In this model, each binaural channel performs feature extraction by analog bandpass filtering and, after that, filtered signals are encoded into events via asynchronous delta modulation (ADM).

Bidimensional (2-D) cochleae model both the propagation of the wave across the basilar membrane and the fluid motion within the cochlea and the basilar membrane. The structure of the filters in this type of artificial cochlea are based in Lyon and Katsiamis' model, which is detailed in section 1.2.3.2.1.3. Neighbor filters are connected through a set of resistors that model the inner fluid of the cochlea. This architecture combines the benefits of the previous 1-D architectures presented before: the coupling of the filters in parallel allows to generate a steep slope in high frequencies despite being second-order filters; the fault-tolerance is improved and the delay accumulation is avoided. In 1992, Watts presented a 50 stages (channels) implementation that improved the dynamic range, stability and errors caused by the transistors (Watts et al., 1992), achieving a uniform frequency response and good values in the quality factors of the filters (Q).

(Fragnière, 1998) describes a 2-D cochlear model in which the pressure and voltage are mathematically analogous to the acceleration of the basilar membrane and the action potentials. Some authors have published recent implementations of this model (Van Schaik and Fragnière, 2001; Shiraishi, 2003; Hamilton et al., 2008; Wen and Boahen, 2009).

The artificial cochleae implemented in the last two works mentioned before (Hamilton et al., 2008; Wen and Boahen, 2009), are active. An artificial cochlea is active when filters change dynamically depending on the input signal. Generally, the gain is increased for low frequencies and decreased for high frequencies. This behavior aims to model the functionality of the outer hair cells (OHCs). These implementations use automatic gain control (AGC) to change the gain of the cochlea depending on the changes produced in the input signal. However, in these cases, the AGC also controls the quality factor (Q) of each section of the cochlea. This way, the gain and the bandwidth of each filter is changed dynamically based on the changes in the characteristics of the input sound.

Table 1.2 compares different analog architectures and models presented in this section.

TABLE 1.2: Summary of characteristics of different analog cochleae.

Reference	No. of sections	Frequency range	Dynamic range	Event rate	Power consumption
(Yang et al., 2016)	64x2	8Hz-20kHz	55 dB	100 kEvents/s.	12-22 mW
(Liu et al., 2014)	64x2	50Hz-50kHz	36 dB	10 MEvents/s.	12-22 mW
(Liu et al., 2010)	64x2	50Hz-50kHz (adjustable)	36 dB	10 MEvents/s.	18-26 mW
(Wen and Boahen, 2009)	360	200Hz-20kHz	52 dB	33 kSpikes/s.	51.8 mW
(Hamilton et al., 2008)	64x2	200Hz-6.6kHz	46 dB	—	56.32 mW

1.3.2.3 Digital implementations

The need for more efficient systems with lesser power consumption and lesser cost led to the research and development of new digital design techniques and architectures. FPGAs (Field-Programmable Gate Array) are a very good option for the development of neuromorphic systems. They are flexible, robust to temperature changes, they have a better dynamic range, a better SNR[6], a simpler computer interface, the board can be used for different applications and their development time is much shorter compared to analog VLSI systems.

The first digital cochlea was implemented in 1992 using an application-specific integrated circuit (ASIC) (Summerfield and Lyon, 1992), Fig. 1.27. It contains 71 sections of cascade filters based on Lyon's cochlear model. The output of each filter is connected to a half-wave rectifier (HWR) that mimics the functionality of OHCs along with an automatic gain control block (AGC). Therefore, even though the model in which it is based is not an active cochlea (it does not have automatic gain control), this digital implementation includes it.

The digital cochlea of (Jones et al., 2000) implemented a second-order band-pass filter bank with 30 filters in an FPGA. This filter bank follows a parallel architecture and it was specifically built for extracting the tone out of complex sounds. Although this digital cochlea has a simple architecture, it implements the model of IHCs and auditory pathways in detail.

In (Leong et al., 2003), an 88-sections cascade was implemented by using second order infinite impulse response filters (IIR). This kind of filters are able to obtain higher and narrower bands with less arithmetic operations. Fig. 1.28 shows the frequency response of some of the sections of the cochlea presented in that work.

[6]SNR, or Signal-to-Noise Ratio, is defined as the ratio of signal power to noise power, often expressed in decibels (dB).

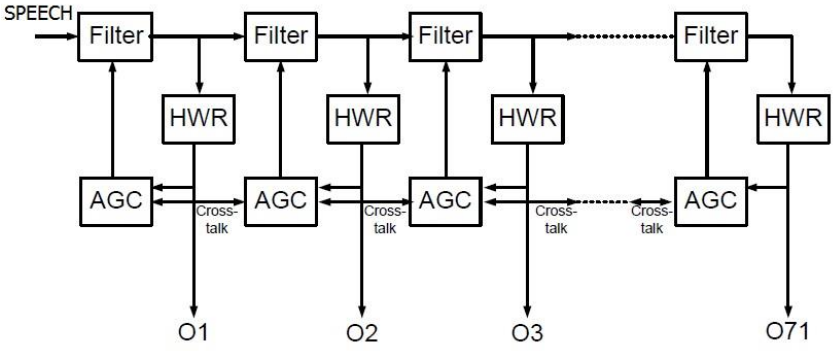
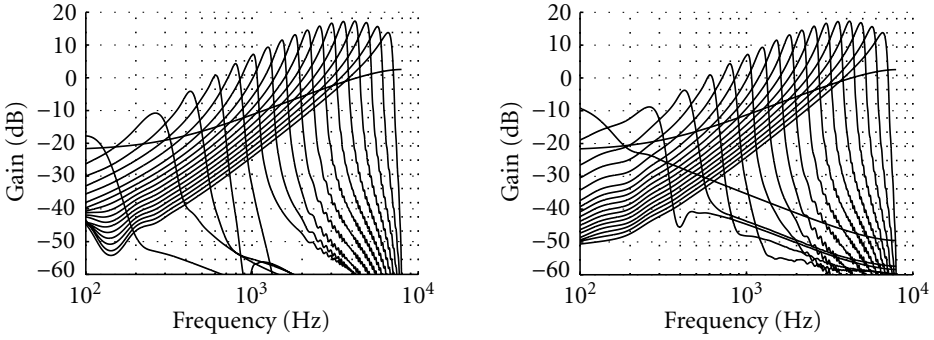


FIGURE 1.27: Block diagram of the digital cochlea proposed by Summerfield et al. Image taken from (Summerfield and Lyon, 1992).



(a) Frequency response (software)

(b) Frequency response(hardware)

FIGURE 1.28: Frequency response of the digital cochlea implementation by Leong et al. Image taken from (Leong et al., 2003).

Parallel architectures achieve a higher speed in the output, but they have a limitation in terms of the cochlea size because the resources needed increase linearly with the number of stages in order to produce the steep slope behavior in the filters. However, in cascade architectures, the design limits are determined by the speed in the output, due to the fact that it decreases linearly with the number of stages.

Some recent implementations of digital cochleae use IIR filters, and both parallel (Dundur et al., 2008; Miró Amarante, 2013) and cascade (Gambin et al., 2010; Mugliette et al., 2011; Thakur et al., 2014) implementations can be found. The digital design presented in (Dundur et al., 2008) is the basis for the cochlear implant implementations in FPGAs.

Other designs (Gambin et al., 2010; Mugliette et al., 2011; Thakur et al., 2014)

use time multiplexing to implement the filter bank with a greater number of sections. For example, the digital cochlea presented in (Mugliette et al., 2011) has a cascade of 24 second-order IIR filters, and thanks to the time multiplexing, it only requires 20 multiplier units. The architecture of this implementation is shown in Fig. 1.29. Designs that are based on time multiplexing use less resources in exchange of using higher clock frequencies, increasing the power consumption of the system.

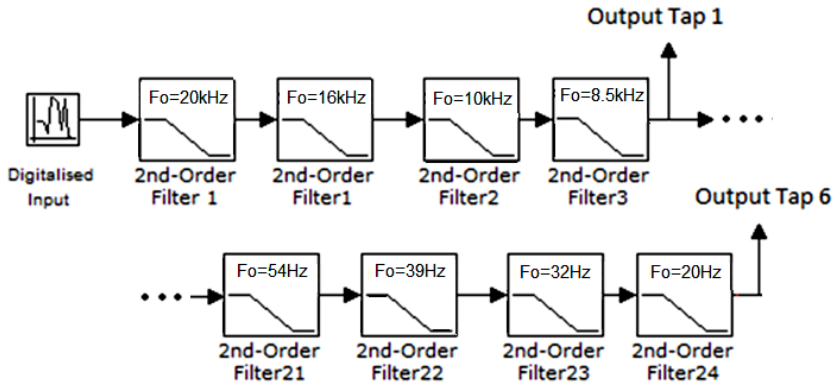


FIGURE 1.29: Filter Cascade with output taps every four sections of the digital implementation proposed by Mugliette et al. Image taken from (Mugliette et al., 2011).

Using FPGA implementations have some advantages over VLSI analog systems: a faster design time, more robustness to power supply changes, to temperatures and to transistor mismatch, a higher dynamic range, a higher SNR, better stability, FPGAs can be reused for different applications and they have a simpler interface to the PC.

Two of the most recent digital cochlea implementations in FPGA are (Jiménez-Fernández et al., 2017) and (Xu et al., 2018). In (Xu et al., 2018), a digital implementation of a 70-section, 44.1 kHz sampling rate Cascade of Asymmetric Resonators with Fast-Acting Compression (CAR-FAC) (Lyon, 2017) cochlear model is presented, where the CAR part simulates the basilar membrane's response to sound and the FAC part models the outer hair cells (OHC), the inner hair cells (IHC), and the medial olivocochlear efferent system functions. The FAC feeds back to the CAR by moving the poles and zeros of the CAR resonators automatically, making it an active cochlea model.

The digital implementation proposed in (Jiménez-Fernández et al., 2017), which is called Neuromorphic Auditory Sensor (NAS), is the one that has been used throughout this thesis. The next section describes this particular implementation.

Table 1.3 compares different digital architectures and implementations presented in this section.

TABLE 1.3: Summary of characteristics of different digital cochleae implementations.

Reference	No. of sections	Frequency range	System clock	Hardware resources
(Xu et al., 2018)	70	Up to 22.05 kHz	250 MHz	49 DSPs 5235 ALMs
(Thakur et al., 2014)	1224	20 Hz - 20.657 kHz	142 MHz	113760 slices 136975 LUTs
(Dundur et al., 2008)	16	150 Hz - 3.4 kHz	—	11048 slices 20699 LUTs
(Leong et al., 2003)	88	1 kHz - 7.630 kHz	6.42 MHz (minimum) 63 MHz (maximum)	5770 slices (minimum) 10771 slices (maximum)

1.3.2.3.1 Neuromorphic Auditory Sensor

Neuromorphic Auditory Sensor (NAS) (Jiménez-Fernández et al., 2017) is an audio sensor for FPGAs inspired by Lyon's model of the biological cochlea (Lyon and Mead, 1988). This sensor is able to process an excitatory audio signal using Spike Signal Processing (SSP) techniques (Jimenez-Fernandez et al., 2010), decomposing incoming audio in its frequency components, and providing this information as a stream of events using the Address-Event Representation (AER) (Boahen, 2000). As it is implemented on a reconfigurable platform, this sensor's configuration parameters are flexible and can be adapted to any application.

NAS decomposes two digitized signals (two in case of a binaural or stereo NAS, or one in case of a monaural or mono NAS) into a set of sections or bands (corresponding to particular frequencies), which are previously converted to a spike train. The decomposition of the signals in each corresponding frequency band is performed by a bank of spike-based SLPF[7] filters (Jimenez-Fernandez et al., 2010; Domínguez-Morales et al., 2011) connected in a cascade topology. The output of this sensor is encoded using the AER address, as many of the neuromorphic sensors that were presented in the previous section.

For this decomposition of the input signals (left and right ear) the same processing is performed. The processing is modeled using a spike-based cascade filter bank (CFB). Each CFB has many stages (as many as sections in the cochlea)

[7]Spike Low-Pass Filter.

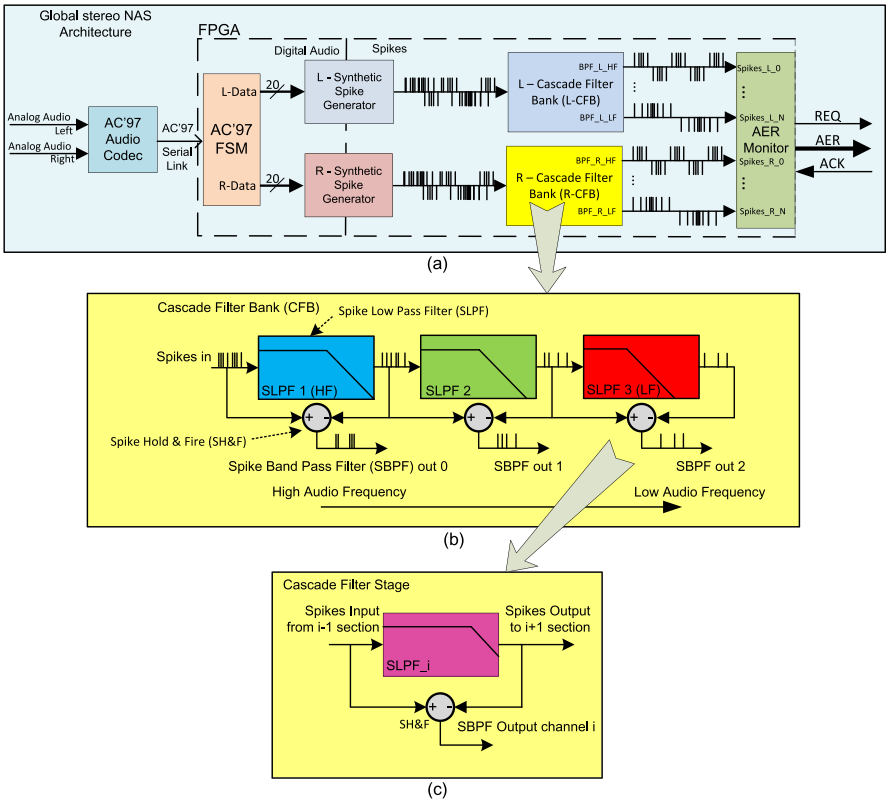


FIGURE 1.30: (a) Global NAS architecture. (b) Filter banks with Cascade topology, CFB. (c) Single CFB stage containing an SLPF and an SH&F. Image taken from (Jiménez-Fernández et al., 2017).

and consists of a SLPF and a SH&F[8] (Jimenez-Fernandez et al., 2010), which are able to extract two different spike trains: one that is passed as input to the next stage and another one that corresponds to the signal filtered by the current stage of the cascade.

This kind of sensors mimic how the biological cochlea processes audio signals. NAS is able to decompose the input audio signal into different frequency bands (also called channels). Like in other neuromorphic cochleae implementations based on a cascade model (Liu et al., 2010; Leong et al., 2003; Summerfield and Lyon, 1992; Lyon and Mead, 1988), this decomposition is carried out by a series of cascade-connected stages that subtract the information from consecutive spike-based low-pass filters' output spikes in order to reject out-

[8]Spike Hold and Fire.

of-band frequencies, obtaining a response equivalent to that of a bandpass filter (Jiménez-Fernández et al., 2017).

Fig. 1.30 (a) shows the global architecture of a binaural NAS. In this architecture, the first element in the chain is an AC97 audio codec (CS5344 audio codec), which has two analog signals as inputs (left and right) that are digitized and multiplexed into a single output. This signal is divided in order to obtain sampled and digitized values that correspond to each of the input signals. After that, the information is converted from a digital domain to a spike domain by using a synthetic spike generator RBSSG (Jimenez-Fernandez et al., 2010), which provides a spike stream with a frequency that is proportional to the digital amplitude. At the output spike generator, ON and OFF spikes that encode the previously sampled digital value are obtained.

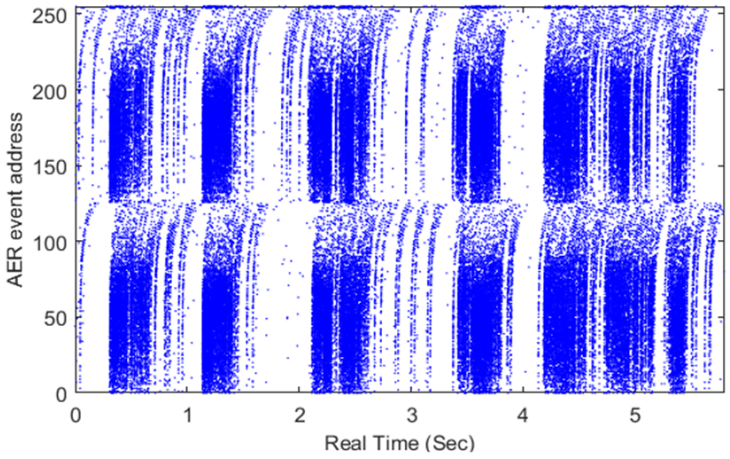


FIGURE 1.31: NAS output representation.

The generated spike train is the input to the cascaded spike-based filter bank CFB (depending on the NAS architecture implemented on the FPGA, the number of filters will be different). Fig. 1.30 (b) shows how these filters are distributed, forming the bank, where the first filters of the cascade correspond to higher frequencies, and the last ones correspond to lower frequencies. As was previously mentioned, the filter bank is divided into different stages, where each stage consists of a spiking low-pass filter along with a SH&F (see Fig. 1.30 (c)). The output of each of the stages is connected to an AER monitor (Cerezuela-Escudero et al., 2013) that encodes the activity of each stage, placing the address of the corresponding stage in the output asynchronous AER bus every time that an event is generated in it. Fig. 1.31 shows NAS's output when it is stimulated by saying the sentence "En un lugar de la Mancha" with a microphone directly connected to the input of the sensor.

TABLE 1.4: Summary of NAS characteristics.

No. of channels	Frequency range	Dynamic range (AC'97 + NAS)	Event rate	Power consumption	System clock	Hardware resources
64x2 (adjustable)	9.6Hz-14.06kHz (adjustable)	75dB	2.19MEvents/s.	29.7mW	27MHz	11141 slices

Table 1.4 presents a summary of NAS characteristics.

1.4 Deep neural networks

Deep neural networks (DNN) are a type of neural network architecture that are able to extract and analyze patterns that are deeply hidden in the input data by using deep learning mechanisms. Deep learning (Deng and Yu, 2014) can be defined as "a class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification".

These algorithms need a vast and diverse labeled dataset to train the network and test its performance. These data provide knowledge to the network in order to develop the tasks they are trained for. Due to the amount of data and high computational cost of this mechanism, it is usually used when simpler methods are insufficient or inadequate for the task that the user aims to accomplish.

Deep neural networks have to learn and classify the whole dataset in an adaptive fashion by using specific training algorithms (Glorot and Bengio, 2010). From (LeCun et al., 2015): "the key aspect of deep learning is that the layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure. Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years".

1.4.1 History

The development of Deep Learning algorithms was one of the main goals of artificial intelligence since its inception, along with one of the main purposes of artificial neural networks (ANN). Using the full computational capabilities of computers to extract more and deeper information than what a human could ever achieve, while applying different mathematical methods to such information, was one of the main ideas behind ANNs. With the continuous advances in computer technology, the necessary computation power has been achieved to perform this kind of algorithms for many different applications with sufficient

speed. Moreover, ANNs are currently being used to model certain parts of the human brain whose functionality is already known (Graupe, 2016).

In 1975, Kunihiko Fukushima (Fukushima, 1975) proposed the Cognitron neural network, which mimics how the retina works for visual pattern recognition. This network was extended in 1982 in a model called Neocognitron (Fukushima and Miyake, 1982), which was still very slow and, like its predecessor, was limited to visual pattern recognition. These neural network architectures are not considered to be part of the Deep Learning field, although they served as a basis for convolutional neural networks, which are the ones that were used as Deep Learning architectures in this work. In 1985, David Rumelhart et al. (Rumelhart et al., 1985) showed that neural networks with many hidden layers could be effectively trained by a simple procedure called Back-Propagation. This would allow neural networks to get around the weaknesses of the perceptron, since the additional layers provided the network with the ability to learn nonlinear functions. BP[9] is based on Richard Bellman's dynamic programming theory (Bellman, 1966) and it is still being used in most Deep Learning architectures.

Convolutional neural networks (CNN) have become the most popular network architecture in the Deep Learning field. Historically, CNNs were inspired by the visual cortex model proposed by Fukushima (Fukushima and Miyake, 1982). This, together with the work carried out by Yann LeCun (LeCun et al., 1989), encouraged the use of CNNs for computer vision problems. In this work from 1989, LeCun included convolutions in his 5-layer model based on BP, achieving faster processing and obtaining more characteristics from the input data, compared to using BP alone. Although these first models took around three days to complete the training phase, current CNN models based on LeCun's LeNet-5 (LeCun et al., 1998), only take several minutes, especially when making use of parallel processing algorithms.

Geoffrey E. Hinton used different CNN models for speech recognition tasks and natural language processing (Hinton et al., 2012), extending their range of applications. CNNs soon became the leading approach for image and audio processing. Currently, CNNs can be used for many more different applications as long as the input information is represented in a two-dimensional space or higher (matrices commonly). Therefore, CNNs have become the most used neural networks for solving Deep Learning complex problems.

1.4.2 Architecture of a Convolutional Neural Network

Convolutional Neural Networks (CNN or ConvNet) are a particular class of deep, feed-forward neural networks where neurons correspond to receptive fields in a

[9] BP, or Backpropagation, is a method used in neural networks to calculate a gradient that is needed for the calculation of the weights to be used in the network.

similar way as the neurons of the primary visual cortex in a brain. Due to this fact, CNNs have been successfully applied to analyze visual imagery. A CNN consists of an input and an output layer, as well as multiple hidden layers. The main difference between CNNs and other different classes of neural networks is the application of convolution operations to extract features from the input images. In addition to these convolution layers, CNNs have different layers that improve and accelerate both the learning and the execution processes by reducing and simplifying the amount of data that is generated. These networks also have other layers dedicated to classification tasks. The most common layers that can be found in a CNN are presented and described in the next section.

1.4.2.1 Convolution layer

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. They are the reason why CNNs are named like that. Convolution layers require a filter (or function) to process the input data. These filters are matrices in which their elements or values are known as weights. These layers consist of many filters, whose values are calculated in the learning phase. Different characteristics obtained from the previous layer are grouped into the so-called "feature maps", which serve as input data to the next layer and are processed using its filters (Equation 1.2), applying convolutions to this information (Equation 1.3) to generate and output. Where Q and R represent the input and output feature maps, respectively.

$$w_{\mathrm{rq}} * x_{\mathrm{q}} = z \tag{1.2}$$

$$z(m,n) = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} w_{\mathrm{rq}}(k,l) * x_{\mathrm{q}}(m+k, n+l) \tag{1.3}$$

The values of each of the filters are calculated in the training phase using the well-known Backpropagation algorithm (LeCun et al., 1998), which consists in propagating the error calculated in the output to every previous layer, adjusting the value of the filters to minimize the error.

Fully connected feedforward neural networks could also be used to extract patterns and learn features, although it is not practical to apply this kind of networks to images because of the number of neurons that would be necessary to process them due to the very large input size. The convolution operation is a solution to this problem, allowing the network to be even deeper while reducing the number of parameters to learn.

1.4.2.2 Pooling layer

This layer combines the outputs of neuron clusters at one layer into a single neuron in the next layer. Pooling layers drastically reduce the width and height of the input feature map, which is why they are also known as downsampling layers. This type of layer has two main purposes: it reduces the amount of weights considerably, lessening the computational cost, and, it decreases the chance to over-fit due to the reduction applied to the spatial information. This layer is not affected by the training phase due to the fact that it does not contain any configurable parameter that needs to be adjusted. In other words, it always applies the same function. The most common pooling functions are:

1.4.2.2.1 Max-pooling

It is the most common function used in this layer and it is usually set as default in most frameworks and tools. It obtains the most representative value from a region of interest, reducing its size. An example of this is shown in Fig. 1.32, where each of the 2x2 regions are replaced with a single value that corresponds to the maximum number inside that region. The size of the pooling kernel (region of interest) is usually set by the user.

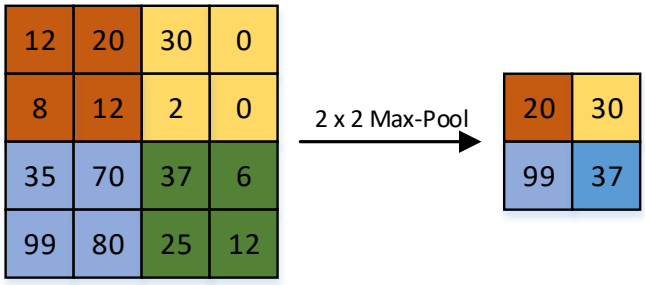


FIGURE 1.32: Max-pooling example.

1.4.2.2.2 Average-pooling

In the average-pooling function (also known as avg-pooling) the arithmetic mean of the values inside the region of interest is computed, instead of using the maximum value. An example of the average-pooling function is shown in Fig. 1.33.

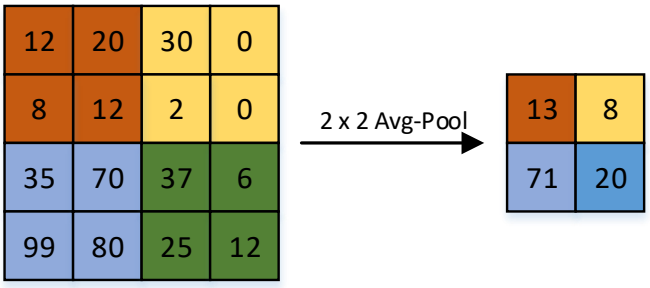


FIGURE 1.33: Average-pooling example.

1.4.2.3 Dropout layer

The idea of the dropout function was first introduced in (Srivastava et al., 2014). It removes a random set of activations in that layer by setting them to zero. This helps reducing the overfitting problem, where the network does not perform well when given new test examples as a consequence of the weights of the network being excessively tuned to the training dataset. The dropout layer is only used in the training phase, and not when testing the network. It is usually used in very deep CNNs that have a high number of feature-maps both in the inputs and in the outputs of the hidden layers. They have not been used in any of the networks that are proposed in this work.

1.4.2.4 Fully-connected layer

The fully-connected layer is also known as the decision or classification phase. It is used to classify the data into various classes with the information obtained from the feature extraction process that was performed in previous layers. The number of outputs in the last fully-connected layer of the CNN corresponds to the number of different classes to be classified.

1.5 SpiNNaker neuromorphic platform

SpiNNaker (Spiking Neural Network Architecture) (Furber et al., 2013) is a massively parallel multicore computing system for modelling very large spiking neural networks in real time, optimized for neuromorphic applications. Both the system architecture and the design of the SpiNNaker chip have been developed by the Advanced Processor Technologies Research Group

(APT)[10], which is based in the School of Computer Science at the University of Manchester. Each SpiNNaker chip consists of eighteen 200 MHz general-purpose ARM968 cores, each with 64kB of tightly-coupled data memory and 32kB of tightly coupled instruction memory. The chip contains a Globally Asynchronous Locally Synchronous (GALS) architecture with an asynchronous packet switching network that is highly optimized for neuromorphic applications (Plana et al., 2007). The communication between them is done via packets carried by a custom interconnect fabric. The transmission of these packets is brokered entirely by hardware, giving the overall engine and extremely high bisection bandwidth. It is important to mention that one of the 18 ARM processors is used for management, and another ARM core is reserved. Therefore, only 16 ARM cores are involved in the neuromorphic process.

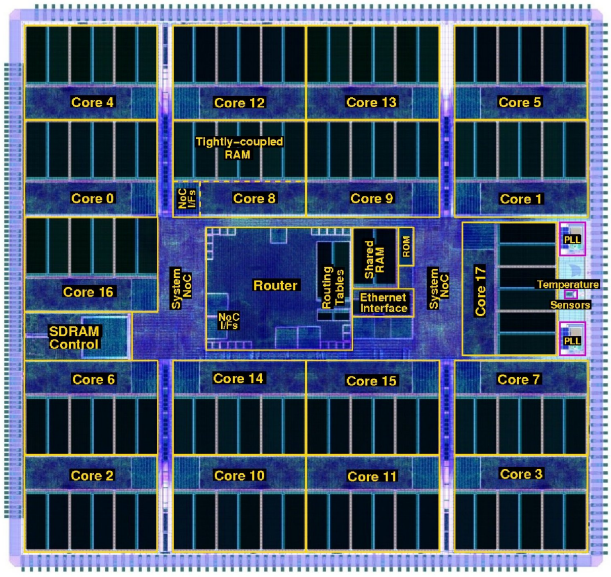


FIGURE 1.34: SpiNNaker chip layout. It contains 18 ARM processors, a Router and SDRAM controller.

Fig. 1.34 shows the layout of the currently available SpiNNaker chip[11]. The ARM cores in a SpiNNaker chip can communicate to each other and to external links through the router represented in Fig 1.34. This router has 24 asynchronous bidirectional links; 18 of them are connected to the 18 ARM processors and six of them are connected to external links. Each processor can host several neurons

[10]The group is based in the School of Computer Science at the University of Manchester and it is headed by Steve Furber. Website: `http://apt.cs.manchester.ac.uk`

[11]SpiNNaker2 is currently under development and in this thesis, we only used the first version of SpiNNaker chip.

and send/receive spikes to other processors through the router and the packet switch network (Plana et al., 2008).

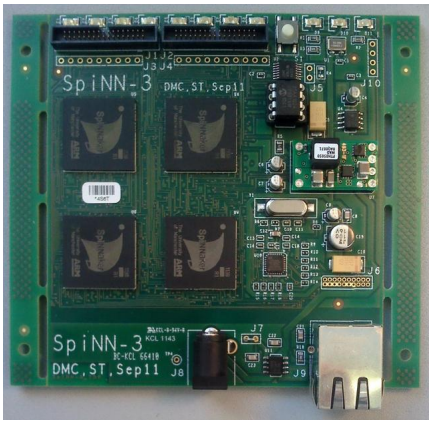


FIGURE 1.35: SpiNN-3 machine.

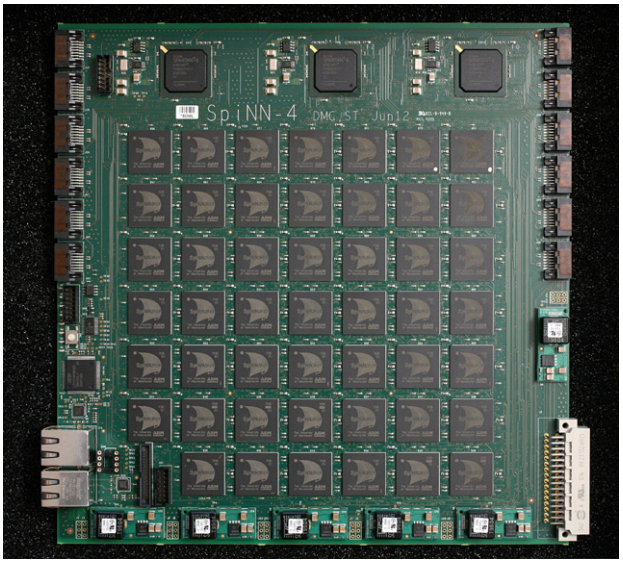


FIGURE 1.36: SpiNN-5 machine.

In this thesis, two different SpiNNaker machines were used: SpiNN-3 (Fig. 1.35) and SpiNN-5 (see Fig. 1.36). The first one is a 4-SpiNNaker-chip circuit board (72 ARM processor cores) and the second one has 48 SpiNNaker chips (86 ARM processor cores) and 3 Spartan-6 FPGAs to communicate to other boards. Both boards have a 100 Mbps Ethernet connection that is used as control and I/O interface between the computer and the SpiNNaker board. SpiNN-5 has

been used to build a million core massively parallel computer for human brain simulation (Furber and Brown, 2009). In this thesis, these boards have been used to deploy large spiking neural networks for audio classification tasks.

Chapter 2

Objectives

"We share the patience of the ages.
Nothing is set in stone."
– The Committee

At the beginning of this work, two types of objectives were proposed: general objectives, with the aim of analyzing and studying the viability of spiking neural networks for audio samples classification and their implementation in real-time neuromorphic hardware platforms, in the context of automatic audio patterns recognition, and more specific objectives, focused on solving particular problems related to the topic introduced previously and the design of the corresponding systems and neural networks.

General objectives: search for and study of new architectures based on spiking systems, analogous to neural systems, for auditory information processing.

1. Study how the human auditory system captures, analyzes and encodes acoustic information into nerve impulses that are processed by the brain.
2. Study neuro-inspired algorithms for training systems to perform a specific task.
3. Develop new systems for processing neuromorphic auditory signals obtained from a NAS.

With this general objective it is intended to mimic the human nervous system to benefit from the evolution of nature, as well as to understand new aspects of how biological neural systems behave. These are the common general objectives of neuromorphic engineering works and developments.

To achieve this general objective, which is broad and ambitious, the following set of specific objectives were proposed:

Specific objectives: implementation and analysis of different mechanisms to perform neuromorphic auditory signals classification based on SNNs and deep learning algorithms.

1. Neuromorphic audio processing:
   (a) Study and analysis of state-of-the-art software tools for processing neuro-inspired auditory information.
   (b) Development of a new application to process this kind of information.
   (c) Development of new tools for generating training datasets for SNNs and CNNs automatically, based on the information received from a NAS.
2. Neuromorphic audio samples classification using spiking neural networks:
   (a) Study of SNNs and how they work.
   (b) Study of different neuro-inspired algorithms for training this kind of network.
   (c) Evaluation and study of different spiking neural network simulators.
   (d) Evaluation and study of the SpiNNaker hardware platform to implement and run SNNs in real time.
   (e) Generation of a training dataset for a first experiment with the aim of performing a classification between different pure tones in a SpiNNaker machine.
   (f) Design of a SNN model to perform the classification.
   (g) Design and modeling of a SNN for real-time pure tones classification.
   (h) Validation and quantification of the obtained results.
3. Neuromorphic audio classification through deep learning mechanisms:
   (a) Study of deep learning mechanisms and CNNs.
   (b) Study of CNNs and how they can be adapted to use neuromorphic auditory information as input.
   (c) Generation of a training dataset for a heart murmur detection system based on CNNs.
   (d) Design and implementation of different CNN models to compare results.
   (e) Analysis of the performance of the system through different tests and comparison of the improvements over other works.
4. Conversion from CNNs to SNNs:

(a) Study of different mechanisms to convert a ANN model to a SNN model.

(b) Study of state-of-the-art algorithms to translate trained weights from a ANN/CNN using backpropagation to the connections of a SNN.

(c) Development of a mechanism for training a CNN off-line and, subsequently, testing of a similar SNN with the weights obtained.

(d) Generation of a training and testing dataset with speech command samples obtained from a NAS.

(e) Performance analysis and deployment of the SNN model in a SpiNNaker machine.

Chapter 3

Summary of results

"My work is my life, my religion and my needs all tied together, for both good and bad."
– Niklas Kvarforth

In this section, the main results and contributions of this thesis are presented. These results are thoroughly described and detailed in two journal papers and two conference contributions that are included as appendices in this thesis.

Different neuromorphic audio classification systems have been developed in this thesis using diverse mechanisms, such as SNN, CNN and a combination of both, including novel pre-processing techniques to adapt the spike-based audio samples to the input of the network and for training these classifiers. To be able to achieve those results, we had to build a software tool for processing the neuromorphic auditory information outputted by the NAS.

Regarding audio processing, there exist many software applications that implement a set of tools for post-processing the information. Audacity, Nero Wave Editor, WavePad, Wavosaur, Ocenaudio, GarageBand and Adobe Audition are some examples of audio editing software. They are able to import and export different audio file formats, trim them, split them into different audio files, apply different effects to the signal and perform a frequency analysis, among other tasks. One of the most popular ones is Audacity, which is free, open-source, cross-platform, easy-to-use and is equipped with an extensive suite of built-in tools, along with a huge selection of third-party plugins to add versatility to this software.

These applications are capable of working with discrete audio samples. However, when it comes to working with spiking information obtained from a neuromorphic sensor, new tools are required for processing this kind of information. jAER (Delbruck, 2008), which is described in section 1.2.3.1., is an open-source software that allows to visualize and record event-based data from diverse neuromorphic sensors, including dynamic vision sensors and artificial cochleae. The fact that jAER accepts a set of different sensors as input makes it a

general purpose neuromorphic software, lacking several specific functionalities that are needed for processing or analyzing auditory information properly.

For this reason, and taking into account that there was no other software capable of processing spike-based audio data, a new application had to be developed to aid neuromorphic engineers on their research in auditory processing. This software is called Neuromorphic Auditory VISualizer (NAVIS) (Dominguez-Morales et al., 2017b). It is free, open-source[1] (under GNU General Public License), easy-to-use and it was created with the idea of having a program analogous to Audacity to work with spike-coded audio information instead of discrete audio samples. This software was published in Neurocomputing journal (3.317 Impact Factor, Q1 Journal Citation Reports), which has created a specific track for Original Software Publications, encouraging researchers in the areas of neural networks and learning systems to publish their software tools.

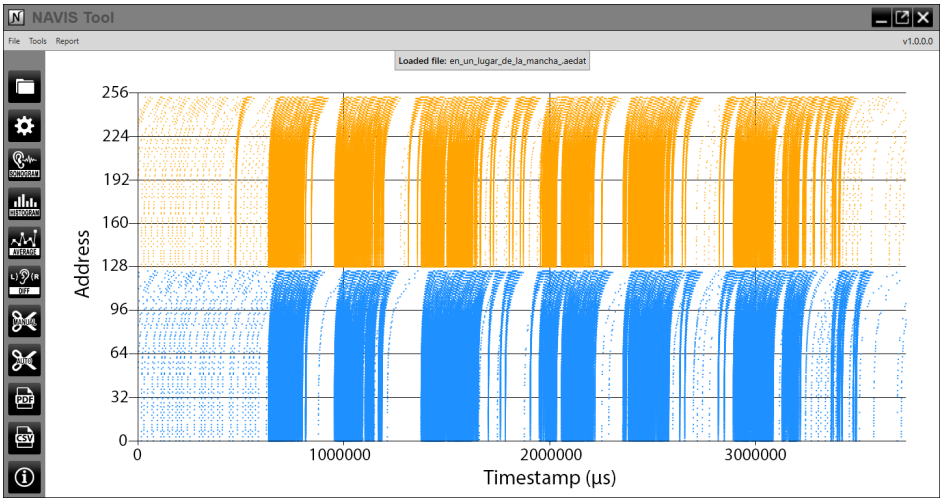


FIGURE 3.1: NAVIS main window showing the cochleogram for the "En un lugar de La Mancha" recording. Left 64-channels represented in blue and right ones in orange.

NAVIS (see Fig. 3.1) is able to read AER streams directly recorded with jAER into AER-data files from a neuromorphic cochlea and perform different functionalities for studying, analyzing and processing spike-encoded information. A data model was designed to store and use the information extracted from AER-data files. This model represents the information efficiently and facilitates its subsequent access and processing. A set of tools were designed to apply operations to the data gathered from the AER-data file and to generate graphical results through its easy-to-use and intuitive graphical user interface (GUI). To improve execution times and reduce resources used in memory,

[1]NAVIS's project and code can be found in https://github.com/jpdominguez/NAVIS-Tool

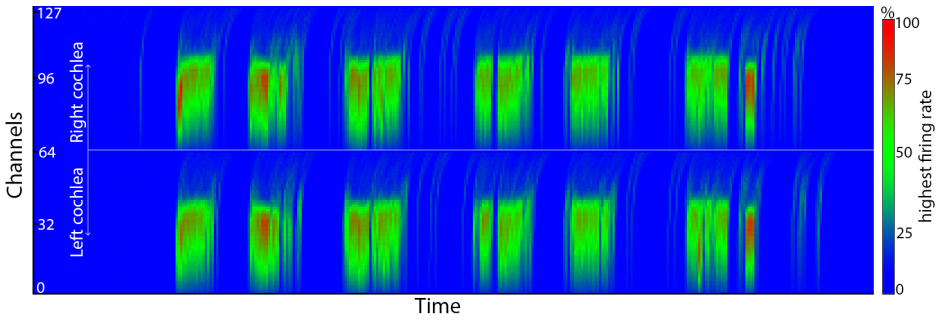


FIGURE 3.2: Sonogram generated with NAVIS. Left 64-channels represented on the bottom part of the image and right ones on top.

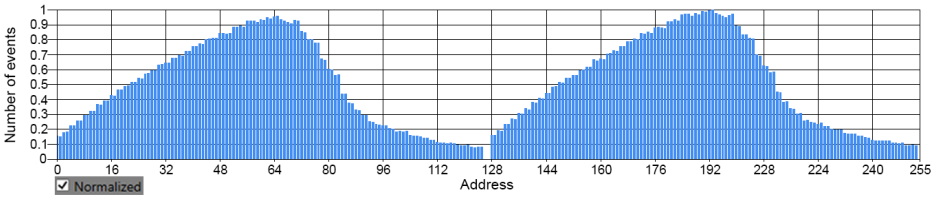


FIGURE 3.3: Histogram generated with NAVIS. Left 128-addresses represented in the left and right ones in the right.

these functions were optimized using LINQ[2] query expressions and lambda functions. To demonstrate these software functionalities, a 64-channel binaural NAS (Jiménez-Fernández et al., 2017) were used together with an USB-AER interface (Berner et al., 2007). The aim of the software is to help the neuromorphic community to work with cochleae data in order to visualize and adapt this information to build training sets for later learning using a neural network or other classifier systems.

Latest NAVIS version (v1.0.1.0) includes the following tools:

- Loading AER-data files generated from AER-based neuromorphic cochleae.
- Cochleogram display (see Fig. 3.1). The cochleogram is an XY graph where X represents timestamp and Y represents AER address. Each of the spikes produced is represented as a dot, determining its address and the time in which it was fired.
- Sonogram generation (see Fig. 3.2). The sonogram represents the spike rate of both cochleae in a color map, where the X axis is time, the Y axis is the NAS channel, and the color is the relative spike rate of the channel for a particular time period.

[2]Language-Integrated Query

- Histogram generation (see Fig. 3.3). For certain applications it is important to know which cochlea channels are the ones that have more activity. The histogram is the appropriate chart to measure this information. In a histogram, the X axis represents the AER address, while the Y axis is the number of events fired.
- Disparity between the left and the right cochleae calculation in an AER-data file obtained from a binaural cochlea. This tool lets the user know which of the two cochleae is predominant at a specific time period in a particular channel.
- Average activity calculation along the length of the file (see Fig. 3.4). This tool generates a 2-axis chart, where the X axis represents time (timestamp, μs) and the Y axis is the number of mega-events (10^6 events) fired per second, calculated by counting the number of events produced between a time period (integration period) and dividing it by this number. This functionality, together with the previous one, could be very useful as an example for echolocation applications.
- Stereo to mono conversion, saving the resultant recording to an AER-data file or to a CSV (comma-separated values) file.
- Mono to stereo conversion, with the possibility of delaying one of the cochleae by a time factor. The resultant information can be saved to an AER-data file or to a CSV file.
- Auditory information splitter and trimmer, both manually and automatically. NAVIS allows to manually extract a specific section of the original file and save it separately. It also implements a splitting function based on the channels' average activity, Fig. 3.5. One of the main aims of NAVIS is to split AER-Data files automatically into different files depending on the fire rate of both channels. A division is established when there is not enough activity to consider a sound, while removing silences. These sections are saved as single AER-Data files. The main goal of this utility is to build training sets for SNNs: after recording a sequence of audio samples into one file, it is able to distinguish all individual stimuli and store them separately.

To test the performance of the Automatic Aedat Splitter tool, a new experiment was carried out. The experiment consisted of automatically splitting an AER-Data file that contained eight different pure tones (different frequencies, 1 second duration each) with a second of silence between them, and then calculating the average error by subtracting the duration of each split from the length that they should have if the pure tone was perfectly extracted from the audio sample. Different integration period values were used; the results can be seen in Fig. 3.6.

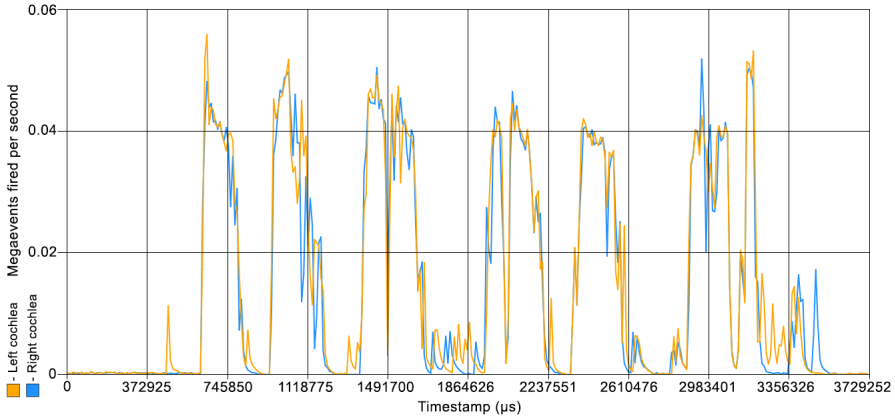


FIGURE 3.4: Average activity of both cochleae.

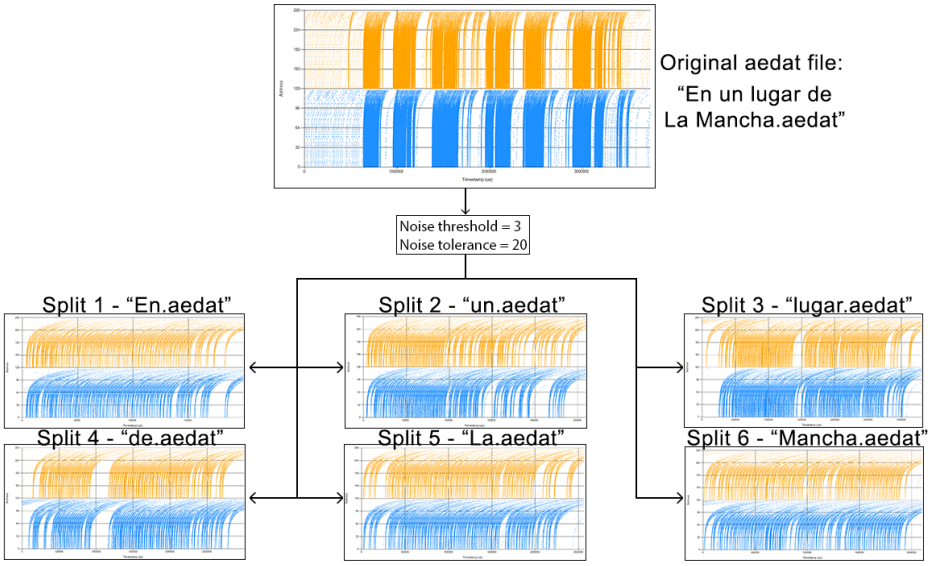


FIGURE 3.5: Automatic AER-Data splitter output.

The main contribution of this software tool is its capability to apply complex audio post-processing treatments and representations, which is a novelty for spike-based systems in the neuromorphic community. This software will help neuromorphic engineers to build sets for the training of spiking neural networks (SNN) thanks to the automatic splitting tool, as well as the rest of the graphical results, Fig. 3.7. NAVIS has served as a basis for processing the auditory information prior to using it for training and testing a neural network. More

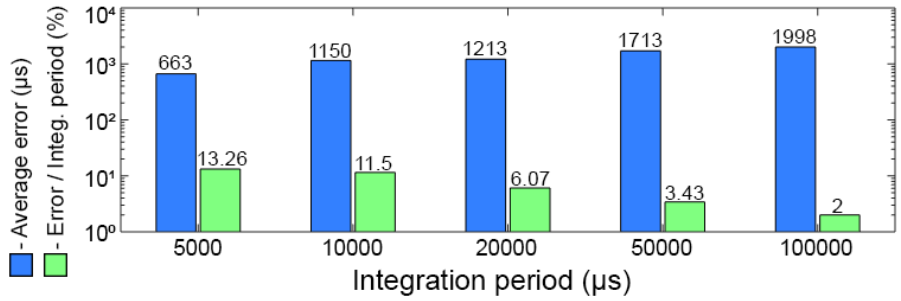


FIGURE 3.6: Average error (μs) and error/integration period percentage obtained when automatically splitting an AER-Data file that contains 8 one-second duration pure tones using different integration period values.

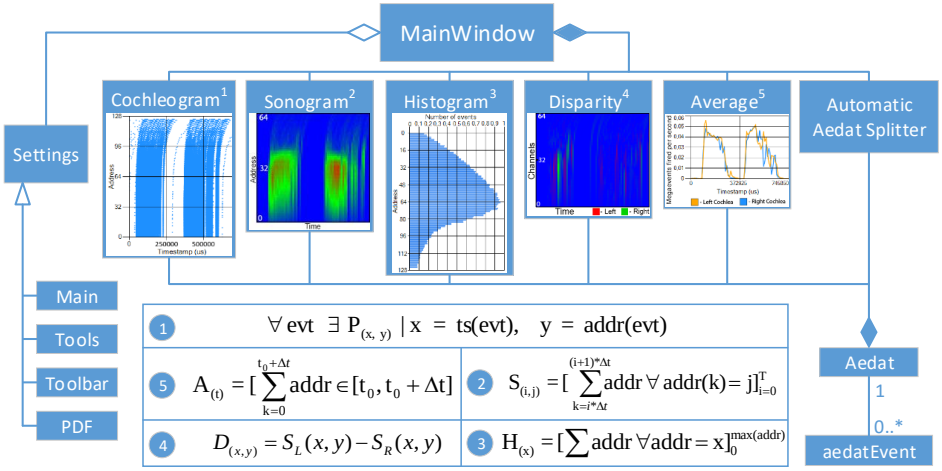


FIGURE 3.7: Block diagram of the overall software architecture describing the algorithms used in the main NAVIS' functionalities.

information about NAVIS and its functionalities can be found in Appendix A. Each of the other papers attached in this thesis (Appendix B, C and **??**) make use of this tool, proving its usefulness within this work.

As a first approach to a neuromorphic audio classification system, a multilayer SNN was built to recognize between eight different pure tones that correspond to specific piano key frequencies (Dominguez-Morales et al., 2016) (Appendix B).

Along with vision, audio classification has always been a research interest within the neuromorphic engineering field. Simulators like Nengo (Bekolay et al., 2014), Brian (Goodman and Brette, 2008) and NEST (Gewaltig and

Diesmann, 2007) (see section 1.2.3.3.) are rapidly increasing in popularity in the neuromorphic community due to the ease of modeling and simulating spiking neural network architectures with them by using a high-level programming language (Python is commonly supported by most SNN simulators). With these frameworks, the user is able to model even complex SNN architectures within a short development time, set the weights and delays of the connections between neurons, describe new cell models and run simulations for a specific time period. However, for interfacing a neuromorphic sensor with the SNN for performing real-time classification or processing tasks based on the spiking information received from the sensor, using these software tools may not be the best option and other different approaches need to be considered.

The SpiNNaker neuromorphic hardware platform (Furber et al., 2013), which is described in section 1.5, allows deploying large SNNs using Python language thanks to sPyNNaker and PyNN, which is a library for SNN description. This board is capable of running SNNs and send reports or results to the computer in real time. Thanks to the external links that the machine has, neuromorphic sensors such as NAS and other hardware devices like FPGAs can be connected to SpiNNaker directly to classify the sensor's information or send information to the external device for further processing.

Matlab's audioplayer function was used to generate 0.5 seconds duration pure tones with a sampling rate of 48 kHz and 1 Vpp. These signals were sent to a 64-channel binaural NAS, generating AER streams that were received in the computer through the USB interface of a USBAERmini2 board that was connected to the NAS. The streams were logged in the computer as AER-data files (with .aedat extension) by using jAER. Fig. 3.8 shows a block diagram of the system.

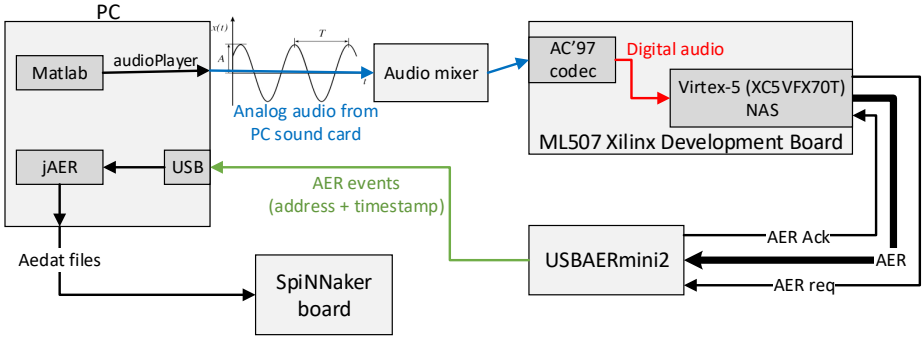


FIGURE 3.8: Block diagram of the pure tones classification system.

A three-layer SNN was built using PyNN and sPyNNaker for later simulation in a SpiNN-3 board, consisting of four SpiNNaker chips and capable

of implementing up to 1024 LIF[3] neurons. The network configuration (see Fig. 3.9) consists of the following components:

- **Input layer**: it receives the stream of AER events that were logged in the computer as AER-data files. This layer consists of 64 neurons due to the fact that the NAS used to process this information is a 64-channel binaural implementation, where the spiking information from the two cochleae is combined.
- **Hidden layer**: this layer represents a first step of the classification of the information contained in the pure tone signals, extracting frequency patterns for further treatment in the output layer. Thus, since the classification consists in recognizing between eight different pure tone signals, this layer consists of eight neurons.
- **Output layer**: eight neurons shape the last layer of the SNN, which improves the pattern extraction from the previous layer, performing the classification and providing the final result.

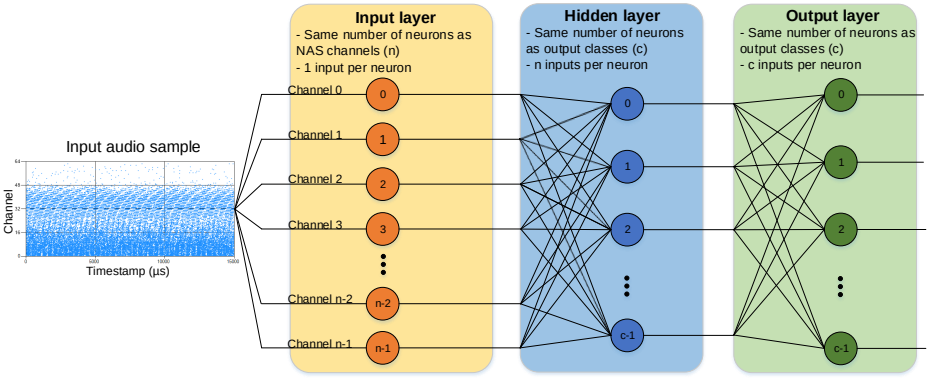


FIGURE 3.9: SNN architecture using an audio sample AER-Data file as input.

A novel offline and supervised firing-rate based algorithm was used to train each connection of the SNN. Two different training steps are performed:

- The weights of the connections from the input layer to the hidden layer are obtained by normalizing the spike firing activity for each NAS channel using a set of audio samples with the same duration, amplitude and frequency as those to be recognized. The firing rate for a specific channel ($FR_{channel\_i}$) (Equation 3.2) is obtained by dividing the number of spikes produced in that channel by the NAS firing rate (FR_{NAS}) (Equation 3.1). This value will be set as the weight for the projection between the

[3]Leaky Integrate-and-Fire

corresponding input neuron and the neuron in the hidden layer that refers to the pure tone that has been used to calculate the firing rate in that particular channel.

$$FR_{\text{NAS}} = (\sum AERevents)/T_{\text{sample}} \tag{3.1}$$

$$FR_{\text{channel\_i}} = (\sum AERevents(i))/FR_{\text{NAS}} \tag{3.2}$$

- The weights of the connections between the hidden layer and the output layer are obtained from the firing output of each neuron in the hidden layer when using the training set of audio samples as inputs to the network after setting the weights that were calculated in the previous step into the connections between the input layer and the hidden layer. These firing outputs are normalized by dividing each of them by the maximum value. These results are the ones that will be used as weights for the connections between the hidden layer and the output layer.

The weights of the connections of the network using these two algorithms were directly obtained from NAVIS after loading each of the AER-data files in the software application.

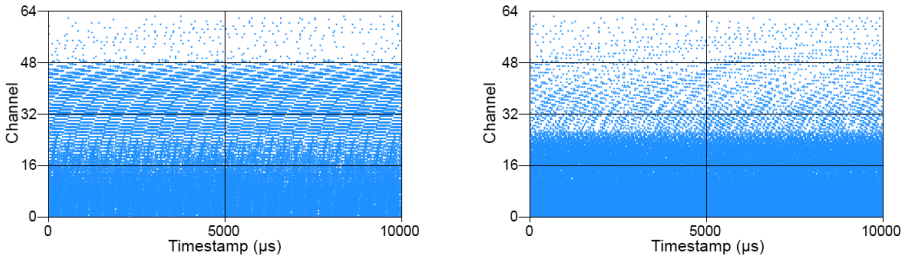


FIGURE 3.10: First 10 ms cochleogram of the 130.813 Hz (left) and 1396.91 Hz (right) pure tones.

In this work, the SNN architecture is trained and tested using eight different pure tones with frequencies that range from 130.813 Hz to 1396.91 Hz, logarithmically spaced (130.813, 174.614, 261.626, 349.228, 523.251, 698.456, 1046.50 and 1396.91 Hz), corresponding to C and F notes from the third to the sixth octave. Fig. 3.10 shows the cochleograms for the 130.813 Hz and the 1396.91 Hz pure tones after logging their spike information in the computer.

Fig. 3.11 shows the normalized spike firing activity for each NAS channel and each of the audio samples used in this experiment.

Different pure tone sets with the same frequency and properties (0.5 s and 0.5 V amplitude) as the ones used to train the network were logged and used to test

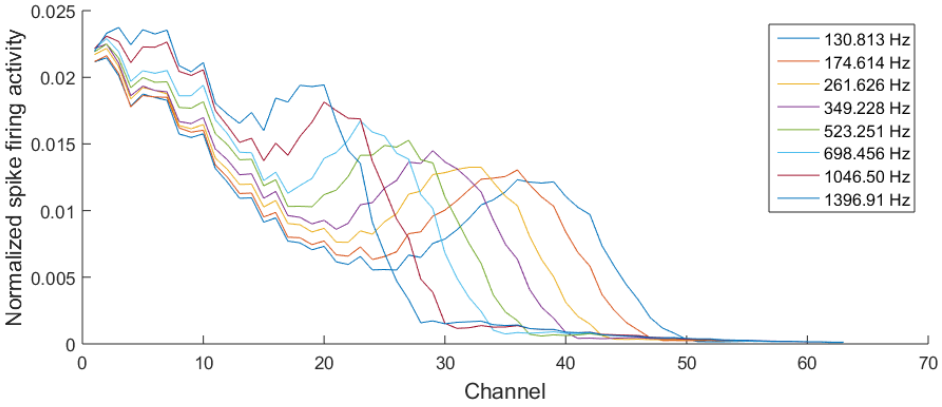


FIGURE 3.11: Normalized spike firing activity for each NAS channel per audio sample.

the network robustness and effectiveness. A 100 % accuracy rate was achieved for every class when the signal was a pure sine wave. Moreover, the network was also tested by adding a random noise signal to the pure tones original signals, obtaining audio samples with different SNR values (from 35.2 dB to 0 dB). Table 3.1 presents the accuracy for every class using the these SNR values.

These results prove the robustness of the SNN-based classification system. Moreover, the authors presented a demonstration of this system at the International Symposium of Circuits and Systems (ISCAS) 2017 in Baltimore (Maryland, USA) (Dominguez-Morales et al., 2017a) with some improvements to the original work, including a NAS to SpiNNaker interface board that allowed the SpiNN-3 board to receive live information directly from the NAS, making the classification work in real time. Some applications for this system could be robotics and automatic musical notes transcription, among others. However, the approach considered in this work regarding the way in which the weights of the connections of the network are set lacks a smart training mechanism, and could not work for more complex problems. This is why new automatic training algorithms were considered after this work. Novel neuromorphic audio processing techniques were used along with well-known deep learning mechanisms for classification tasks with a more relevant application in our daily life such as the eHealth field. In particular, the next experiment was focused on the detection of heart murmurs within heart sound recordings in order to determine whether a person is healthy or not.

The heart is one of the most important organs in the human body and is the one affected by a larger number of diseases[4]. Around 17.7 million people die

[4]Diseases that affect the heart are called Cardiovascular Diseases or CVDs.

TABLE 3.1: Accuracy of the pure tones classification SNN for different SNR values.

SNR	**Pure tone frequency (Hz)**							
(dB)	**130.813**	**174.614**	**261.626**	**349.228**	**523.251**	**698.456**	**1046.5**	**1396.91**
No noise	100%	100%	100%	100%	100%	100%	100%	100%
35.1993	100%	100%	100%	100%	100%	100%	100%	100%
21.3363	100%	83%	96%	100%	100%	100%	100%	100%
13.2273	100%	81%	92%	100%	100%	100%	100%	96%
7.4733	100%	86%	100%	100%	100%	100%	100%	95%
3.0103	74%	90%	100%	98%	100%	100%	100%	98%
2	93%	88%	20%	32%	16%	92%	32%	97%
1	10%	5%	0%	0%	0%	88%	26%	94%
0	0%	0%	0%	0%	0%	76%	22%	91%

each year from cardiovascular diseases (CVD), which represent 31% of all deaths worldwide (World Health Organization, 2017). Detecting CVDs at an early stage is crucial for applying the corresponding treatment and reducing the potential risk factors. Auscultation is one of the most used techniques for this purpose, and can provide clues to the diagnosis of many cardiac abnormalities by listening and analyzing the heart sound components using a stethoscope. It is very cheap and requires minimal equipment. However, physicians need extensive training and experience for auscultating (Roy et al., 2002). Moreover, the accuracy rate of primary care physicians and medical students on the auscultation process is between 20% and 40%, as reported in (Etchells et al., 1997; Mangione and Nieman, 1997; Lam et al., 2005; Strunic et al., 2007), and only roughly 80% is achieved by expert cardiologists (Etchells et al., 1997; Strunic et al., 2007; Ejaz et al., 2004).

Heart murmurs are the most common abnormal finding when a patient visits the physician for auscultation. Based on the experience in the auscultation process, the physician must decide whether the patient is healthy or not; however, due to the fact that the accuracy is not very high, the expert could be wrong, making type-I or type-II errors. A type-I error is the detection of an effect that is not present (i.e. healthy patients are sent for echocardiogram), whereas a type-II error is the failure to detect an effect that is present (i.e., pathological patients are sent home without medication or treatment). It is clear that, in this case, type-II errors are more important to avoid. However, echocardiograms cost between $750 and $1500 (Etchells et al., 1997) per patient, which makes type-I errors also important to consider and avoid. The probability of needing this costly procedure could be reduced for both healthy people and pathological patients if a reliable diagnostic tool were available as an aide for physicians.

Studying heart murmurs and developing mechanisms to detect them is not

a novel topic. Many studies have worked towards designing practical murmur classifier systems to improve the diagnostic accuracy of physicians. Most of them use ANNs, support vector machines (SVM) or some complex preprocessing algorithms (Strunic et al., 2007; Ejaz et al., 2004; Rios-Gutierrez et al., 2012; Hadi et al., 2008; Hadi et al., 2010; Jia et al., 2012; Singh and Cheema, 2013; Leung et al., 2000; Noponen et al., 2007; Markaki et al., 2013; Perera et al., 2013; Potes et al., 2016; Zabihi et al., 2016). However, these approaches present different downsides: some of them have only trained the network with simulated heart sounds without noise, obtaining bad accuracy results when testing the classifier with real heart sounds (Strunic et al., 2007; Rios-Gutierrez et al., 2012; Hadi et al., 2008; Hadi et al., 2010), others have used only a small amount of real heart sounds for training the system, lacking robustness in a real scenario (Hadi et al., 2010; Jia et al., 2012; Leung et al., 2000; Noponen et al., 2007; Markaki et al., 2013; Perera et al., 2013), and others have presented non-automated solutions, having a preprocessing step where a person selects the best portion of the heart sound signal to be used as input to the system (Hadi et al., 2008; Hadi et al., 2010; Noponen et al., 2007). A more detailed study of the literature regarding heart sound detection and recognition is presented in Appendix C.

These downsides and approaches were analyzed in order to build a novel heart murmur detection system to improve current state-of-the-art solutions. This was achieved by pre-processing heart sound recordings in a bio-inspired way using a NAS and performing the classification with different CNN models trained with deep learning algorithms. The heart sound recordings dataset was obtained from the PhysioNet/CinC Challenge 2016 (Goldberger et al., 2000), which consists of 3126 heart sound recordings, ranging from 5 to over 120 seconds in length. Each of these sounds were first segmented (see Fig. 3.12 (a)) using three different segmentation windows of 1, 1.25 and 1.50 seconds duration, obtaining three different datasets: 77573 1-second-long audio samples, 61518 1.25-second-long audio samples, and 51009 1.5-second-long audio samples. Each of the samples from each dataset were sent to a 64-channel monaural NAS, whose output was logged to the computer using a USBAERmini2 board and Matlab into AER-Data files (see Fig. 3.12 (b)). Fig. 3.12 shows the NAS (implemented in the AER-Node platform) connected to the USBAERmini2. With the spiking information contained in these files, and by using NAVIS, one sonogram image was generated for each of the recordings (using time windows of 20 ms in length for integrating the information) (see Fig. 3.12 (c)). Thus, three different sets of images were obtained: 77573 images with a size of 50x64 for the 1 second-long samples dataset, 61518 images with a size of 63x64 for the 1.25 second-long samples dataset, and 51009 images with a size of 75x64 for the 1.5 second-long samples dataset. Different CNN models were trained with these images in order to improve the accuracy obtained in state-of-the-art approaches performed by other authors. The default LeNet-5 (LeCun et al., 1998) and AlexNet (Krizhevsky et al., 2012) CNN models, along with two modified versions of them, were trained and tested with the three different datasets. More details about the

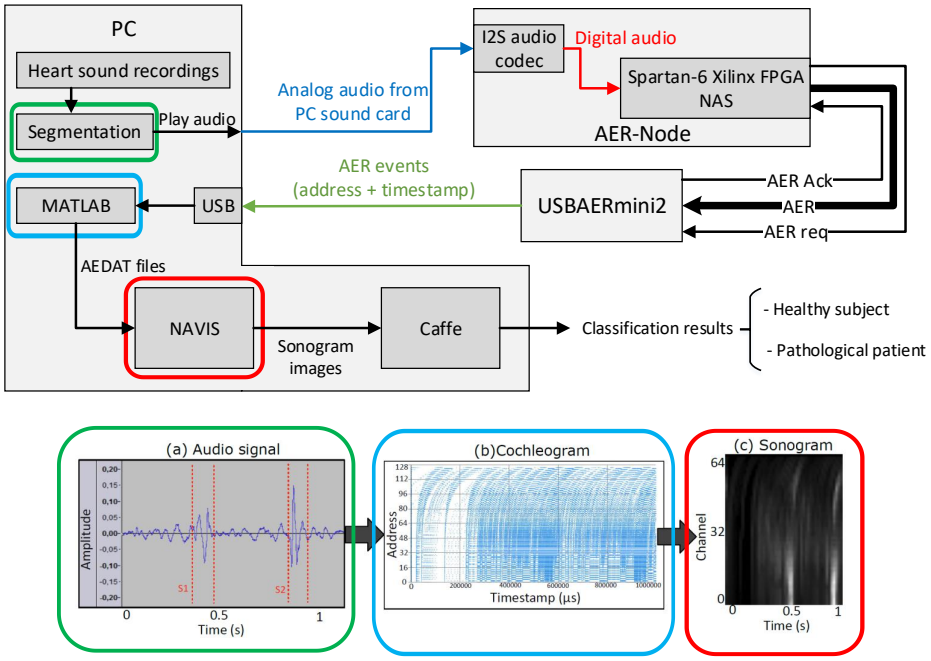
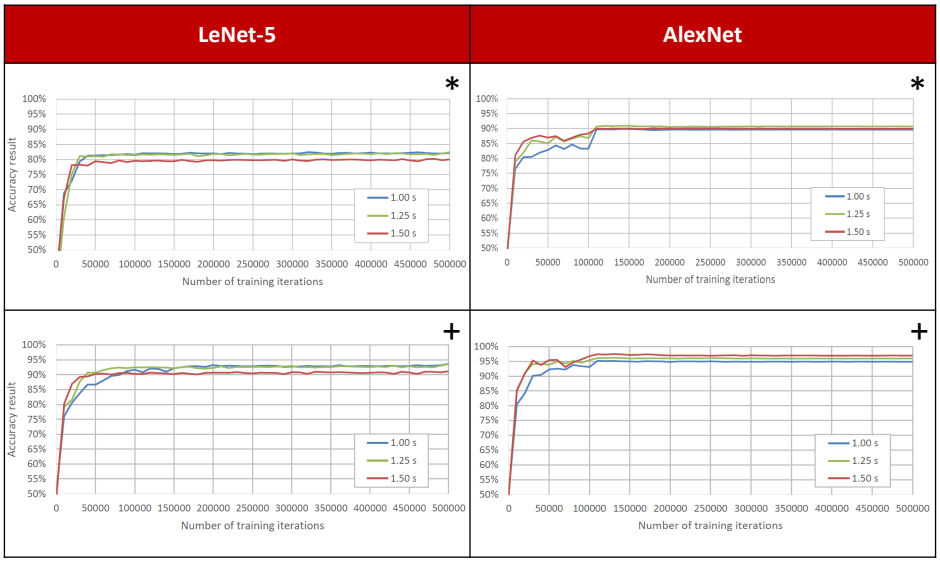


FIGURE 3.12: Block diagram of the architecture of the heart murmurs detection system and outputs of the different preprocessing steps.

modifications that were performed to the default models can be seen in Appendix C (Dominguez-Morales et al., 2018a).

Fig. 3.2 shows the evolution of the training process through time for each of the CNNs and each of the datasets. The best result was obtained with the modified AlexNet model and the 1.5 second-long samples, achieving a sensitivity of 95.12% and a specificity of 93.20%. Table 3.3 presents the best results obtained per CNN model compared to the ones obtained in other papers. As can be seen, the results achieved in this work improve current state-of-the-art solutions in terms of accuracy, making it a useful tool to aid physicians in the auscultation process. These results were achieved thanks to the use of a bio-inspired pre-processing step using a NAS, which is also almost 4 times less power consuming than implementing a Fast-Fourier Transform (FFT) for audio frequencies decomposition, as other previous works have done: a 64-channel binaural NAS has a power consumption of 29.7 mW (Jiménez-Fernández et al., 2017) and, on the other hand, implementing the FFT in an FPGA consumes around 125 mW (Mookherjee et al., 2015). We also have to consider the fact that we are using a 64-channel monaural NAS, with half the number of filters with respect to the binaural version, which means that the power consumption is even lower than 29.7 mW.

TABLE 3.2: Accuracy results achieved for each dataset (1s in blue, 1.25s in green and 1.5s in red) per 10000 training iterations using the four different CNN models.



The results for the heart murmur detection system were obtained in Caffe Deep Learning Framework (Jia et al., 2014). This framework cannot interface directly with NAS and, thus, this classification cannot be done in real time as it is. In (Dominguez-Morales et al., 2018a), a hardware setup was proposed to convert this system into a CNN-based neuromorphic stethoscope. However, using a deep SNN instead of a deep CNN alternative can provide a machine learning system with power saving and input noise tolerance benefits (Farabet et al., 2012). Additionally, such deep SNNs can be trained on input data generated from a neuromorphic spiking sensor device, unlocking the potential for a real-time inference system on a spiking neuromorphic platform (Perez-Peña et al., 2017). Therefore, in order to make a full neuromorphic classification system using SNNs instead of CNNs, a new experiment was carried out, with the aim of classifying voice commands in order to command a robot.

Voice commands are commonly used in multiple personal virtual assistants (Cooper et al., 2004), like Cortana in Microsoft Windows, or Siri in iOS. Users are able to control their personal computers or mobile phones by using natural language sentences. This kind of assistants are based on a field of Artificial Intelligence (AI) called Natural Language Processing (NLP) to identify what the user is saying (Chowdhury, 2003; Gazdar and Mellish, 1989).

TABLE 3.3: Accuracy, sensitivity, specificity and PhysioNet/CinC Challenge 2016 score of the different studied approaches. Best cases for the 1, 1.25 and 1.5 datasets are selected.

	Accuracy	**Sensitivity *(Se)***	**Specificity *(Sp)***	***MAcc***
Primary care physicians	40%	-	-	-
Expert cardiologists	80%	-	-	-
(Potes et al., 2016)	-	94.24%	77.81%	0.8602
(Zabihi et al., 2016)	-	86.91%	84.90%	0.8590
Default LeNet-5	82.39%	83.26%	78.58%	0.8092
Modified LeNet-5	93.68%	92.84%	91.48%	0.9216
Default AlexNet	90.70%	94.52%	90.48%	0.9250
Modified AlexNet	97.05%	95.12%	93.20%	0.9416

In recent years, the application of Artificial Neural Network (ANN) to this field has become commonplace. Notably, the combination of Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) has led to significant progress in developing human-machine interface, as in (Williams and Renals, 1997; McGuire et al., 2002; Russakovsky et al., 2015; Collobert et al., 2016).

There exist several frameworks and training mechanisms to train CNNs, making this process a relatively easy task, as can be seen from the previous work. The most used training algorithm (for ANN and CNN training) is the well-known Levenberg-Marquardt back-propagation algorithm (Hagan and Menhaj, 1994). In contrast, there is no established standard training algorithm for SNNs.

Spike-Timing-Dependent Plasticity (STDP) is a biological process that is able to adjust the strength (weights) of the connections between neurons based on the relative timing of a particular neuron's output and input spiking activity. This process has been implemented in several simulators and hardware platforms, including SpiNNaker, and has become one of the most ubiquitous approaches for training spike-based networks especially for unsupervised learning (Diehl and Cook, 2014). STDP has proved to be very useful and robust for static input signals such as images (Iakymchuk et al., 2015; Kheradpisheh et al., 2016), although it is more difficult to apply when it comes to processing time-varying signals such as audio samples.

As an alternative to STDP, the weights of the connections between neurons in a network could be set by hand or based on particular statistical algorithms, as was done in Appendix B (Dominguez-Morales et al., 2016). This option is complex because it generally needs several trial-and-error loops in order to find the best weight configuration, which can take a long time. Also, this way of

setting the weights of the connections is too task specific and lacks the generality and biological plausibility of STDP.

In recent years, the difference in classification error between deep SNNs and deep ANNs has diminished significantly (Liu and Furber, 2016). These exciting results suggest that, if trained appropriately, a SNN can be used for machine learning inference without introducing penalties in data classification accuracy. The general off-line SNN training method proposed in (Liu et al., 2017) is based on two novel activation functions: Noisy Softplus (NSP) (Liu and Furber, 2016), which closely mimics the LIF firing activity driven by current influx with different noise levels, and Parametric Activation Function (PAF), which maps abstract numerical numbers of activation functions to specific physical units of a spiking neuron. Thus, the combination provides an equivalent representation of a spiking LIF neuron with abstract activation functions of ANNs. PAF allows using more generalized activation functions (e.g., ReLU instead of NSP) to model a LIF neuron once its parameters are fitted by NSP. Therefore, the weights of a SNN can be trained off-line on an equivalent ANN exactly the same way as conventional ANNs (e.g., using Backpropagation and Stochastic Gradient Descent), although using PAFs.

Based on this approach, an off-line SNN training tool containing the Matlab code for ANN training and the Python code for reading trained weights, translating into PyNN language and testing the SNN in NEST, was used to build a neuromorphic speech command recognition system. The "left" and "right" voice commands from the Speech Command dataset, which consists of 65000 one-second long utterances of 30 short words, were used, since one of the final goals of the COFNET project is to drive a robot using these two voice commands in a neuromorphic way. This dataset has 4720 "left" and "right" audio files from thousands of different speakers. Each of the audio samples were sent to the audio input of a 32-channel monaural NAS. An USBAERmini2 board receives NAS's output and sends it to the computer through the USB, where it is logged using a MATLAB script into AER-data files (see Fig. 3.13). These files were then converted to sonogram images using NAVIS's algorithms (see Appendix A) (Dominguez-Morales et al., 2017b). To do this, a bin width of 20 ms was selected in order to calculate the firing rate for each of the NAS's channels in every bin. In order to make the training of the network more robust to a real scenario, in which the core information of the audio could be presented not only in the center of the image but in any position of it, an overlapping shifting window was used, generating several images for each audio sample with the information centered in different timestamps. A total of 141726 images were generated in this process, 121565 of which were used to train the network and the remaining 20161 images to test the accuracy of the system. Sonogram images from "left" and "right" speech commands can be seen in Fig. 3.14.

A 5C-3P-3C-2P Spiking Convolutional Neural Network (5x5 kernel-size convolutional layer followed by a 2x2 pooling layer, another 3x3 convolutional

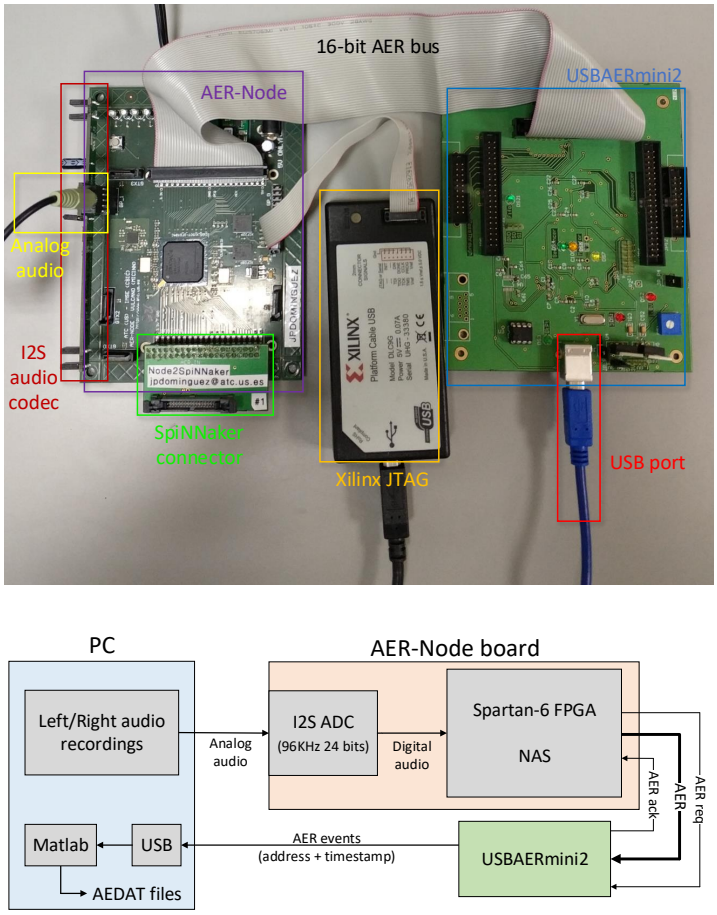


FIGURE 3.13: Picture (top) and block diagram (bottom) of the hardware setup for the dataset generation.

layer followed by a 2x2 pooling layer, and then a fully connected layer) was trained in Matlab with the rate-based sonograms described before. An accuracy result of 92.21% was achieved when training the CNN for 30 epochs, at a learning rate value of 0.1 and a synaptic time constant of 0.005 ms, using the ReLU activation function on the fully connected layer. After this, the network was fine-tuned for one more epoch with the Noisy Softplus function. After the fine-tuning process, the network obtained 90.80% accuracy; although this is a slightly lower value, it will improve the performance when translating from the ANN in Matlab to a SNN in pyNN (NEST).

The weights obtained from the training step were saved and used to build a SNN with the same architecture as the ANN that was trained. The SNN was built

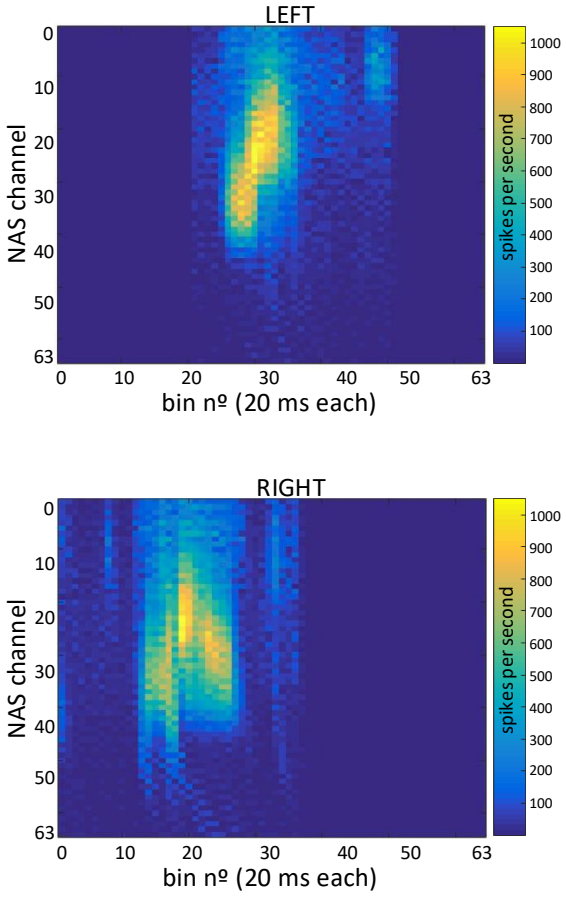


FIGURE 3.14: Sonogram images corresponding to one "Left" (top) and one "Right" (bottom) audio samples from the Speech Command dataset after obtaining their spiking information from the NAS.

using PyNN and, after that, it was simulated in NEST. Then, the same SNN was deployed in a SpiNN-5 machine (with 48 SpiNNaker chips), achieving 89.90% accuracy. The confusion matrix is shown in Fig. 3.15.

The aforementioned developments in deep SNNs show accurate classification of static input data (images) using a deep convolutional SNN. This work shows that it is possible to train a similarly structured network on time series input data from a NAS produced from a range of sound inputs by generating a training dataset consisting of many overlapping 'snapshots' of the NAS output.

To make this classification in real time by using the NAS-to-SpiNNaker

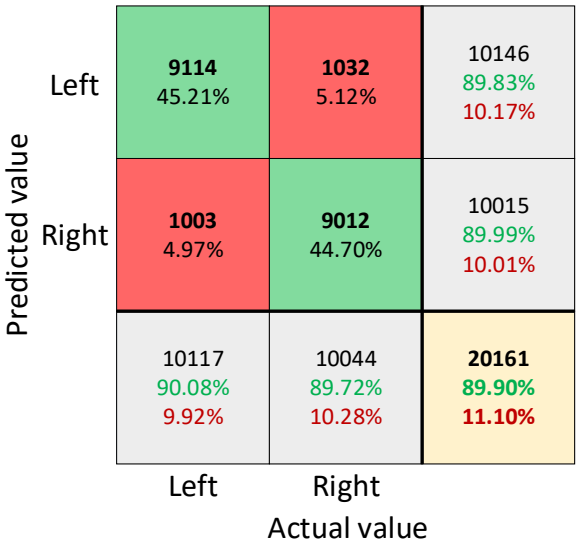


FIGURE 3.15: Confusion matrix of the SNN test using 20161 samples (10117 "left" and 10044 "right" samples).

interface that was presented in (Dominguez-Morales et al., 2017a), Appendix **??** (Dominguez-Morales et al., 2018b) presents and describes a novel SNN architecture for neuromorphic audio samples classification using the output from a NAS as input to the network and a buffering layer with delayed populations that adapts the information from a real-time domain to a static domain, in which the SNN is trained for, based on the SNN training method that was used before. This approach could also be used for processing time series or time-dependent signals with SNNs in real time. This SNN architecture for real-time classification can be seen in Fig. 3.16, and a detailed description on how the information is processed within it can be found in Appendix **??**.

To sum up, this thesis has presented different novel contributions to the neuromorphic engineering field and, in particular, to the neuromorphic audio processing field. First, a desktop software tool for post-processing neuromorphic auditory information obtained from a NAS was developed. This software serves as a basic tool for the rest of the contributions presented in this thesis, which are different audio classification systems. The first system is used to classify between eight different pure tone signals using a SNN implemented in SpiNNaker and a NAS, and was extended in a demonstration for ISCAS 2017 (Dominguez-Morales et al., 2017a), improving the system and making it work in real time. Then, a deep-learning-based approach of the heart murmur recognition problem was developed using neuromorphic audio processing thanks to the use of a NAS, improving current state-of-the-art solutions and becoming a useful tool

to aid cardiologists and primary care physicians in the auscultation process. To conclude, a deep-learning-based training system was built in Matlab using specific activation functions that allow to train a CNN, fine-tune the weights obtained and export them to a SNN without modifying them, achieving almost the same accuracy in the training phase and when testing the SNN. This framework was used to train a network for "left" and "right" speech commands recognition, which was later deployed on a SpiNNaker board, obtaining good accuracy results. A novel SNN architecture is proposed to classify time-variant signals such as speech commands in real time.

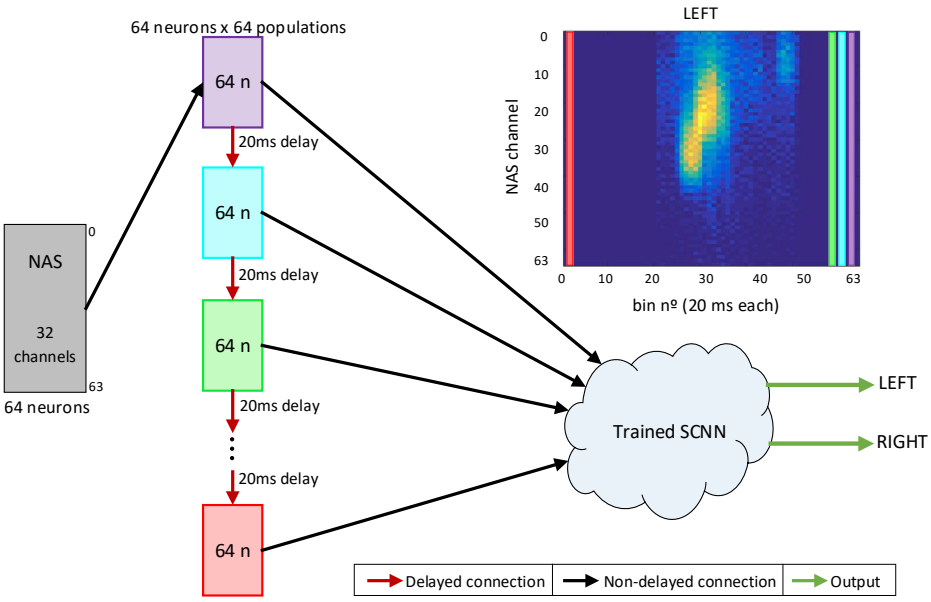


FIGURE 3.16: Real-time NAS audio input SCNN scenario with a buffering layer consisting of a set of delayed populations.

Chapter 4

Discussion

"Nothing lasts forever but the certainty of change."
– Bruce Dickinson

In the previous section, the results obtained in this thesis were presented. In this chapter, the main contributions achieved are described comparing them to other state-of-the-art approaches, along with future works that could be investigated considering the works presented in this thesis as a starting point.

The development of NAVIS has led to a very important and useful tool that has been completely necessary for performing the other works that are part of the compendium of contributions presented in this thesis. Until the release of NAVIS, there was no specific tool for analyzing and processing raw information obtained from a NAS. Moreover, the functionalities that it has for generating training datasets will certainly be used in future works. Being free and open-source makes NAVIS easy to be used, shared and modified by anyone who wants to use it within the neuromorphic community. NAVIS is constantly evolving and improving, with new tools and functionalities for neuromorphic auditory information analysis and processing being added in each update, along with fixing minor bugs reported by the users.

With the help of this tool and the SpiNNaker platform, an SNN-based off-line pure tone classifier system was built. This was the first time that NAS and SpiNNaker were used together, achieving what we believe is the first ever neuromorphic auditory classifier implemented on this hardware platform. Thanks to the layer configuration and firing-rate-based algorithms to set the weights of the connections, the noise tolerance achieved was high, improving previous results (Cerezuela-Escudero et al., 2015) even when using a simpler network configuration. This work was improved with a real-time NAS-to-SpiNNaker interface, making the classification run in real time. This setup could be used in future works for more complex real-time neuromorphic audio signals classification (e.g. music transcription). With this implementation, we aim to model the inferior colliculus of the brain, which is the point in the brainstem where all auditory pathways traveling through the brainstem converge. It

receives information from the cochleae and the superior olivary nuclei and processes it before it reaches the cerebral cortex. The cells of the central nucleus of the inferior colliculus are organized tonotopically, meaning that different neurons respond preferentially to different sound frequencies. It is also involved in fine-tuning auditory sensations and it might be responsible for pitch detection.

From the inferior colliculus, the information travels to the cortex. The visual cortex consists of a set of cells that are responsible to detect borders and orientations within visual stimuli; this is, they are able to extract information from the spatial domain and the relation between adjacent receptive fields. Taking into account these fundamentals of biology, heart murmur recordings were processed with a NAS and the spiking information was integrated using 20 ms bins, generating sonogram images where the spatial information in the X axis represents time. Although heart murmurs detection is not a novel topic, this work was the first to use neuromorphic audio processing, which is over four times less power consuming than using an FFT (Mookherjee et al., 2015). This pre-processing phase, along with the training and fine-tuning of different CNN models led to a heart murmur detection system that improved state-of-the-art solutions to the problem. The leading approach from PhysioNet/CinC challenge 2016 (Goldberger et al., 2000; Potes et al., 2016) was overcome by the work presented in this thesis (Appendix C) by more than 1% sensitivity and 6% specificity, meaning that this approach is the current best option in terms of accuracy for aiding physicians in the auscultation process. Although real-time detection is not a constraint for this specific application, a hardware setup was proposed in the paper in order to make this approach work in real time (neuromorphic stethoscope), Fig. 4.1.

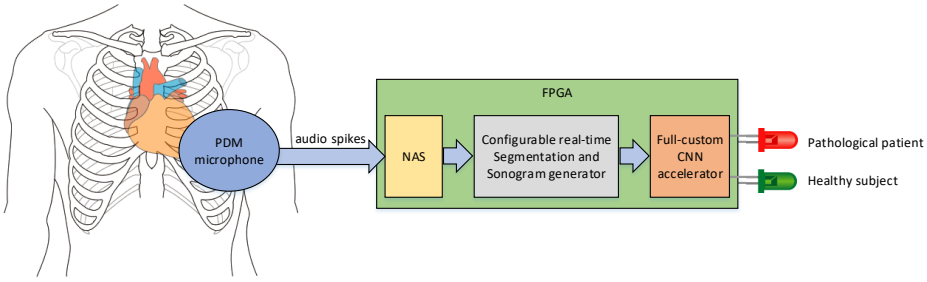


FIGURE 4.1: Block diagram of the complete system implemented on an FPGA using a PDM microphone for real-time analysis of the heart sound directly from the patient.

It was also achieved in this work the first neuromorphic speech command recognition implemented in SpiNNaker using a neuromorphic sensor. The knowledge obtained from previous works regarding SNNs, CNNs, neuromorphic audio processing and the SpiNNaker itself were put together

in order to develop a framework for off-line training SNNs based on deep-learning mechanisms. The system was tested and validated, obtaining 90% accuracy for "left" and "right" spoken commands classification. In Appendix **??**, a novel SCNN model was proposed for performing the classification in real time, along with the necessary mechanisms to train this network with any dataset of spike-encoded information, although we focused on time-variant signals as audio. Based on this real-time neuromorphic speech commands classification in SpiNNaker, we would like, as a future work, to increase the number of spoken words that the system is able to classify and command a four-wheeled robot with them using the interface between SpiNNaker and FPGA that was built in (Dominguez-Morales et al., 2017a). As was previously described, this framework could also be used for training SNNs with any kind of data that can be represented as an image, becoming an easier and faster option for training this kind of network as opposed to using STDP.

To conclude, this thesis has presented novel contributions in the field of neuromorphic audio processing both in real-time and off-line using different neural networks such as SNNs, CNNs and SCNNs, which have been implemented in embedded neuromorphic hardware platforms (SpiNN-3 and SpiNN-5). Some of the results improve previous state-of-the-art solutions (Appendices A and C), and others are implementations that have been developed for the first time (Appendices A, B and **??**), becoming important contributions that have been published in high impact factor journals and conferences. Some lines of research have been opened with these results, allowing numerous possibilities for future works, some of which have been presented in this discussion. Fig. 4.2 shows a block diagram with the relations and connections of the papers included as appendices in this thesis.

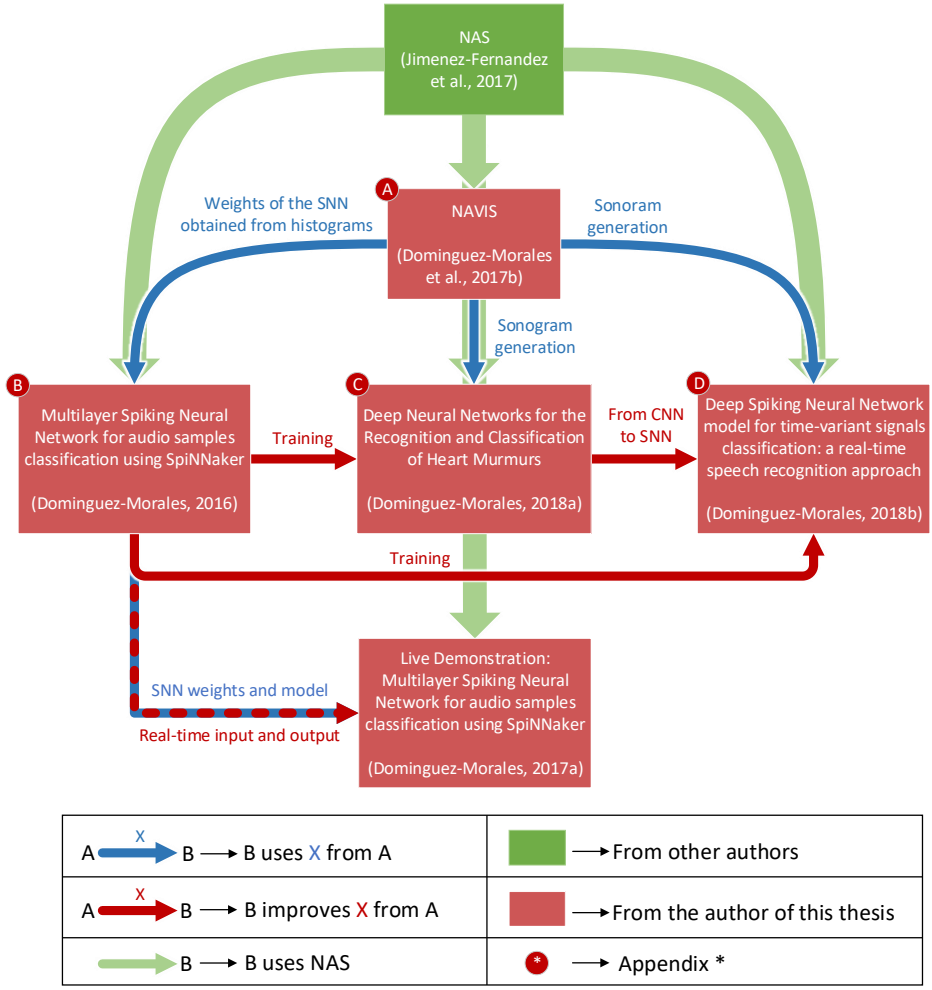


FIGURE 4.2: Block diagram of the connections between the papers attached as appendices in this thesis.

Chapter 5

Conclusions

"In my last mourning I realize,
I've always been the nothing I feared becoming."
– Paolo Bruno

In this work, which has been presented throughout this document, the following contributions and conclusions are highlighted:

- An in-depth study of the human auditory system and the most relevant models of the cochlear system behavior has been carried out.
- A study of the functioning of biological neurons and the connections between them has been conducted, along with an analysis of the codification of spike-based information.
- Current software tools and utilities for neuromorphic auditory information processing have been studied and analyzed. The advantages and disadvantages have been taken into account to develop a novel software application called NAVIS for processing spike-encoded information obtained from a NAS. Moreover, some new tools have been added with the aim of generating datasets automatically for training a network. This software was published in the Neurocomputing journal, presented in the Capo Caccia Neuromorphic Cognitive Neuromorphic Engineering Workshop 2017, and it is free, open and public for the neuromorphic engineering community, both for using it and programming new functionalities.
- The SpiNNaker neuromorphic hardware has been studied along with the necessary Python libraries to work and communicate with the machines. This knowledge, along with NAVIS, has been used to develop a pure tone classification system based on an SNN that receives recorded spiking information from a NAS. The system has been implemented on a SpiNN-3 machine, where the performance was analyzed with noisy inputs, achieving good accuracy in, what we believe it is, the first implementation of a neuromorphic audio classifier to use this hardware platform. This

work was presented in the International Conference on Artificial Neural Networks (ICANN) 2016, and published in Lecture Notes in Computer Science (LNCS) as a book chapter.

- A study of deep-learning mechanisms and convolutional neural networks has been performed. This knowledge has been used to develop a heart murmur detection system based on CNNs and spiking audio information logged from a NAS. The audio samples were obtained from a public dataset for the PhysioNet/CinC Challenge 2016, which were processed using NAS and NAVIS in order to generate an image dataset. Four different CNN models were trained and tested, achieving very good results. These results were compared to the leading approaches from the challenge and other state-of-the-art works, obtaining the best results in terms of accuracy, specificity and sensitivity. This work was published in the IEEE Transaction on Biomedical Circuits and Systems journal and presented in the International Symposium on Circuits and Systems (ISCAS) 2018.
- During the three-month research internship in the Advanced Processor Technologies of the University of Manchester, which is headed by Steve Furber, a framework was developed for off-line training SNNs with deep-learning algorithms from a CNN for neuromorphic audio information classification. This system was tested with a "left" and "right" speech commands dataset. The classifier was deployed in a SpiNN-5 machine in, what we believe it is, the first implementation of a neuromorphic speech commands recognition system using to use this approach and a SpiNNaker hardware platform. The results have been presented in the International Joint-Conference on Neural Networks (IJCNN) 2018 and published in the conference proceedings.
- A novel SNN model for real-time neuromorphic audio samples classification has been designed using the output from a NAS as input to the network and a buffering layer with delayed populations that adapts the information from a real-time domain to a static domain based on an off-line training based on deep-learning algorithms.

Chapter 6

Bibliography

"As if all this was something more than another footnote on a postcard from nowhere. Another chapter in the handbook for exercises in futility."

– Mikołaj Żentara & Maciej Kowalski

Barbancho, Ana M, Anssi Klapuri, Lorenzo J Tardón, and Isabel Barbancho (2012). "Automatic transcription of guitar chords and fingering from audio". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.3, pp. 915–921.

Barlow, Horace B (1961). *Possible principles underlying the transformations of sensory messages*. MIT press.

Bascuas, Luís Enrique López (1997). "La percepción del habla: problemas y restricciones computacionales". In: *Anuario de psicología/The UB Journal of psychology* 72, pp. 3–22.

Bekolay, Trevor, James Bergstra, Eric Hunsberger, Travis DeWolf, Terrence C Stewart, Daniel Rasmussen, Xuan Choo, Aaron Voelker, and Chris Eliasmith (2014). "Nengo: a Python tool for building large-scale functional brain models". In: *Frontiers in neuroinformatics* 7, p. 48.

Bellman, Richard (1966). "Dynamic programming". In: *Science* 153.3731, pp. 34–37.

Berge, Hans Kristian Otnes and Philipp Hafliger (2007). "High-speed serial AER on FPGA". In: *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE, pp. 857–860.

Berner, Raphael, Tobi Delbruck, A Civit-Balcells, and Alejandro Linares-Barranco (2007). "A 5 Meps $100 USB2.0 address-event monitor-sequencer interface". In: *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE, pp. 2451–2454.

Boahen, Kwabena A (2000). "Point-to-point connectivity between neuromorphic chips using address events". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47.5, pp. 416–434.

Braitenberg, Valentino and Almut Schüz (1998). "Cortical architectonics". In: *Cortex: Statistics and Geometry of Neuronal Connectivity*. Springer, pp. 135–137.

Cant, Nell B and Christina G Benson (2003). "Parallel auditory pathways: projection patterns of the different neuronal populations in the dorsal and ventral cochlear nuclei". In: *Brain research bulletin* 60.5-6, pp. 457–474.

Cerezuela-Escudero, Elena, Manuel Jesus Dominguez-Morales, Angel Jiménez-Fernández, Rafael Paz-Vicente, Alejandro Linares-Barranco, and Gabriel Jiménez-Moreno (2013). "Spikes monitors for FPGAs, an experimental comparative study". In: *International Work-Conference on Artificial Neural Networks*. Springer, pp. 179–188.

Cerezuela-Escudero, Elena, Angel Jimenez-Fernandez, Rafael Paz-Vicente, M Dominguez-Morales, Alejandro Linares-Barranco, and Gabriel Jimenez-Moreno (2015). "Musical notes classification with neuromorphic auditory system using FPGA and a convolutional spiking network". In: *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, pp. 1–7.

Cerezuela-Escudero, Elena, Fernando Pérez-Peña, Rafael Paz-Vicente, Angel Jimenez-Fernandez, Gabriel Jimenez-Moreno, and Arturo Morgado-Estevez (2018). "Real-time neuro-inspired sound source localization and tracking architecture applied to a robotic platform". In: *Neurocomputing* 283, pp. 129 –139. ISSN: 0925-2312.

Chan, Vincent, Shih-Chii Liu, and Andr van Schaik (2007). "AER EAR: A matched silicon cochlea pair with address event representation interface". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.1, pp. 48–59.

Chan, Vincent Yue-Sek, Craig T Jin, and André van Schaik (2012). "Neuromorphic audio-visual sensor fusion on a sound-localising robot". In: *Frontiers in neuroscience* 6, p. 21.

Chowdhury, Gobinda G (2003). "Natural language processing". In: *Annual review of information science and technology* 37.1, pp. 51–89.

Collobert, Ronan, Christian Puhrsch, and Gabriel Synnaeve (2016). "Wav2letter: an end-to-end convnet-based speech recognition system". In: *arXiv preprint arXiv:1609.03193*.

Cooper, Robert S, Jeff F McElroy, Walter Rolandi, Derek Sanders, Richard M Ulmer, and Edward Peebles (2004). *Personal virtual assistant*. US Patent 6,757,362.

Davison, Andrew P, Daniel Brüderle, Jochen M Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger (2009). "PyNN: a common interface for neuronal network simulators". In: *Frontiers in neuroinformatics* 2, p. 11.

Delbruck, Tobi (2008). "Frame-free dynamic digital vision". In: *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, pp. 21–26.

Deng, Li, Dong Yu, et al. (2014). "Deep learning: methods and applications". In: *Foundations and Trends® in Signal Processing* 7.3–4, pp. 197–387.

Diehl, Peter U and Matthew Cook (2014). "Efficient implementation of STDP rules on SpiNNaker neuromorphic hardware". In: *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, pp. 4288–4295.

Ding, Qian and Nian Zhang (2007). "Classification of recorded musical instruments sounds based on neural networks". In: *Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on*. IEEE, pp. 157–162.

Dominguez-Morales, Juan P, A Rios-Navarro, D Gutierrez-Galan, R Tapiador-Morales, A Jimenez-Fernandez, E Cerezuela-Escudero, M Dominguez-Morales, and A Linares-Barranco (2017a). "Live demonstration—Multilayer spiking neural network for audio samples classification using SpiNNaker". In: *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, pp. 1–1.

Dominguez-Morales, Juan P, Angel Jimenez-Fernandez, M Dominguez-Morales, and Gabriel Jimenez-Moreno (2017b). "NAVIS: Neuromorphic Auditory VISualizer Tool". In: *Neurocomputing* 237, pp. 418–422.

Dominguez-Morales, Juan P, Angel F Jimenez-Fernandez, Manuel J Dominguez-Morales, and Gabriel Jimenez-Moreno (2018a). "Deep Neural Networks for the Recognition and Classification of Heart Murmurs Using Neuromorphic Auditory Sensors". In: *IEEE transactions on biomedical circuits and systems* 12.1, pp. 24–34.

Dominguez-Morales, Juan Pedro, Angel Jimenez-Fernandez, Antonio Rios-Navarro, Elena Cerezuela-Escudero, Daniel Gutierrez-Galan, Manuel J Dominguez-Morales, and Gabriel Jimenez-Moreno (2016). "Multilayer spiking neural network for audio samples classification using SpiNNaker". In: *International Conference on Artificial Neural Networks*. Springer, pp. 45–53.

Dominguez-Morales, Juan Pedro, Qian Liu, Robert James, Daniel Gutierrez-Galan, Angel Jimenez-Fernandez, Simon Davidson, and Steve Furber (2018b). "Deep Spiking Neural Network model for time-variant signals classification: a real-time speech recognition approach". In: *International Joint-Conference on Neural Networks*. IEEE, pp. 45–53.

Domínguez-Morales, M, Angel Jimenez-Fernandez, Elena Cerezuela-Escudero, Rafael Paz-Vicente, Alejandro Linares-Barranco, and Gabriel Jimenez (2011). "On the designing of spikes band-pass filters for FPGA". In: *International Conference on Artificial Neural Networks*. Springer, pp. 389–396.

Dundur, Rekha V, MV Latte, SY Kulkarni, and MK Venkatesha (2008). "Digital filter for cochlear implant implemented on a field-programmable gate array". In: *proceedings of world academy of science, engineering and technology*. Vol. 33. Citeseer.

Ejaz, Khaled, Glenn Nordehn, Rocio Alba-Flores, Fernando Rios-Gutierrez, Stanley Burns, and Nicholas Andrisevic (2004). "A heart murmur detection system using spectrograms and artificial neural networks." In: *Circuits, Signals, and Systems*, pp. 374–379.

Engineer, Crystal T, Claudia A Perez, YeTing H Chen, Ryan S Carraway, Amanda C Reed, Jai A Shetake, Vikram Jakkamsetti, Kevin Q Chang, and Michael P Kilgard (2008). "Cortical activity patterns predict speech discrimination ability". In: *Nature neuroscience* 11.5, p. 603.

Etchells, Edward, Chaim Bell, and Kenneth Robb (1997). "Does this patient have an abnormal systolic murmur?" In: *Jama* 277.7, pp. 564–571.

Farabet, Clément, Rafael Paz, Jose Pérez-Carrasco, Carlos Zamarreño, Alejandro Linares-Barranco, Yann LeCun, Eugenio Culurciello, Teresa Serrano-Gotarredona, and Bernabe Linares-Barranco (2012). "Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel ConvNets for visual processing". In: *Frontiers in neuroscience* 6, p. 32.

Fletcher, Harvey (1940). "Auditory patterns". In: *Reviews of modern physics* 12.1, p. 47.

Fletcher, Harvey and Wilden A Munson (1933). "Loudness, its definition, measurement and calculation". In: *Bell Labs Technical Journal* 12.4, pp. 377–430.

Fragnière, Eric (1998). *Analogue VLSI emulation of the cochlea*.

Fujii, Hiroshi, Hiroyuki Ito, Kazuyuki Aihara, Natsuhiro Ichinose, and Minoru Tsukada (1996). "Dynamical cell assembly hypothesis—theoretical possibility of spatio-temporal coding in the cortex". In: *Neural Networks* 9.8, pp. 1303–1350.

Fukushima, Kunihiko (1975). "Cognitron: A self-organizing multilayered neural network". In: *Biological cybernetics* 20.3-4, pp. 121–136.

Fukushima, Kunihiko and Sei Miyake (1982). "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, pp. 267–285.

Furber, Stephen and Andrew Brown (2009). "Biologically-inspired massively-parallel architectures-computing beyond a million processors". In: *Application of Concurrency to System Design, 2009. ACSD'09. Ninth International Conference On*. IEEE, pp. 3–12.

Furber, Steve (2016). "Large-scale neuromorphic computing systems". In: *Journal of neural engineering* 13.5, p. 051001.

Furber, Steve B, David R Lester, Luis A Plana, Jim D Garside, Eustace Painkras, Steve Temple, and Andrew D Brown (2013). "Overview of the spinnaker system architecture". In: *IEEE Transactions on Computers* 62.12, pp. 2454–2467.

Furber, Steve B, Francesco Galluppi, Steve Temple, and Luis A Plana (2014). "The spinnaker project". In: *Proceedings of the IEEE* 102.5, pp. 652–665.

Gambin, Isabel, Ivan Grech, Owen Casha, Edward Gatt, and Joseph Micallef (2010). "Digital cochlea model implementation using Xilinx XC3S500E spartan-3E FPGA". In: *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*. IEEE, pp. 946–949.

Gazdar, Gerald and Christopher S Mellish (1989). *Natural language processing in Lisp: An introduction to computational linguistics*. Addison-Wesley Wokingham, England.

Gelfand, Stanley A (2017). *Hearing: An introduction to psychological and physiological acoustics*. CRC Press.

Gewaltig, Marc-Oliver and Markus Diesmann (2007). "Nest (neural simulation tool)". In: *Scholarpedia* 2.4, p. 1430.

Glasberg, Brian R and Brian CJ Moore (1990). "Derivation of auditory filter shapes from notched-noise data". In: *Hearing research* 47.1-2, pp. 103–138.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256.

Goldberger, Ary L, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley (2000). "Physiobank, physiotoolkit, and physionet". In: *Circulation* 101.23, e215–e220.

Gómez-Rodríguez, F, A Jiménez-Fernández, F Pérez-Peña, L Miró, MJ Domínguez-Morales, A Ríos-Navarro, E Cerezuela, D Cascado-Caballero, and A Linares-Barranco (2016). "ED-Scorbot: A robotic test-bed framework for FPGA-based neuromorphic systems". In: *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on*. IEEE, pp. 237–242.

Goodman, Dan FM and Romain Brette (2008). "Brian: a simulator for spiking neural networks in Python". In: *Frontiers in neuroinformatics* 2, p. 5.

Graupe, Daniel (2016). *Deep Learning Neural Networks: Design and Case Studies*. World Scientific Publishing Company.

Hadi, Haslina Mohamed, Mohd Yusoff Mashor, Mohd Zubir Suboh, and Mohamed Sapawi Mohamed (2010). "Classification of heart sound based on s-transform and neural network". In: *Information Sciences Signal Processing and their Applications (ISSPA), 2010 10th International Conference on*. IEEE, pp. 189–192.

Hadi, HM, MY Mashor, MS Mohamed, and KB Tat (2008). "Classification of heart sounds using wavelets and neural networks". In: *Electrical Engineering, Computing Science and Automatic Control, 2008. CCE 2008. 5th International Conference on*. IEEE, pp. 177–180.

Hagan, Martin T and Mohammad B Menhaj (1994). "Training feedforward networks with the Marquardt algorithm". In: *IEEE transactions on Neural Networks* 5.6, pp. 989–993.

Hamilton, Tara Julia, Craig Jin, Andre Van Schaik, and Jonathan Tapson (2008). "An active 2-D silicon cochlea". In: *IEEE Transactions on biomedical circuits and systems* 2.1, pp. 30–43.

Hines, Michael L and Nicholas T Carnevale (2001). "NEURON: a tool for neuroscientists". In: *The neuroscientist* 7.2, pp. 123–135.

Hinton, Geoffrey, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97.

Hodgkin, Alan L and Andrew F Huxley (1939). "Action potentials recorded from inside a nerve fibre". In: *Nature* 144.3651, p. 710.

Hull, Kerry L (2011). *Human Form, Human Function: Essentials of Anatomy & Physiology*. Lippincott Williams & Wilkins.

Hynna, Kai and Kwabena Boahen (2001). "Space-rate coding in an adaptive silicon neuron". In: *Neural Networks* 14.6-7, pp. 645–656.

Iakymchuk, Taras, Alfredo Rosado-Muñoz, Juan F Guerrero-Martínez, Manuel Bataller-Mompeán, and Jose V Francés-Víllora (2015). "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification". In: *EURASIP Journal on Image and Video Processing* 2015.1, p. 4.

Indiveri, Giacomo, Elisabetta Chicca, and Rodney Douglas (2006). "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity". In: *IEEE transactions on neural networks* 17.1, pp. 211–221.

Jäckel, David, Rico Moeckel, and Shih-Chii Liu (2010). "Sound recognition with spiking silicon cochlea and Hidden Markov Models". In: *Ph. D. Research in Microelectronics and Electronics (PRIME), 2010 Conference on*. IEEE, pp. 1–4.

Janus, Scott (2004). *Audio in the 21st Century*. Intel Press.

Jesus Guerrero-Turrubiates, Jose de, Sheila Esmeralda Gonzalez-Reyna, Sergio Eduardo Ledesma-Orozco, and Juan Gabriel Avina-Cervantes (2014). "Pitch estimation for musical note recognition using artificial neural networks". In: *Electronics, Communications and Computers (CONIELECOMP), 2014 International Conference on*. IEEE, pp. 53–58.

Jia, Lijuan, Dandan Song, Linmi Tao, and Yao Lu (2012). "Heart sounds classification with a fuzzy neural network method with structure learning". In: *International Symposium on Neural Networks*. Springer, pp. 130–140.

Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell (2014). "Caffe: Convolutional architecture for fast feature embedding". In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, pp. 675–678.

Jimenez-Fernandez, Angel, Alejandro Linares-Barranco, Rafael Paz-Vicente, Gabriel Jiménez, and Antón Civit (2010). "Building blocks for spikes signals processing". In: *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, pp. 1–8.

Jimenez-Fernandez, Angel, Gabriel Jimenez-Moreno, Alejandro Linares-Barranco, Manuel J Dominguez-Morales, Rafael Paz-Vicente, and Anton Civit-Balcells (2012). "A neuro-inspired spike-based PID motor controller for multi-motor robots with low cost FPGAs". In: *Sensors* 12.4, pp. 3831–3856.

Jiménez-Fernández, Angel, Elena Cerezuela-Escudero, Lourdes Miró-Amarante, Manuel Jesus Domínguez-Morales, Francisco de Asís Gómez-Rodríguez, Alejandro Linares-Barranco, and Gabriel Jiménez-Moreno (2017). "A binaural neuromorphic auditory sensor for FPGA: a spike signal processing approach". In: *IEEE transactions on neural networks and learning systems* 28.4, pp. 804–818.

Johnston, Daniel and Samuel Miao-Sin Wu (1994). *Foundations of cellular neurophysiology*. MIT press.

Jones, Simon, Ray Meddis, Seow Chuan Lim, and A Robert Temple (2000). "Toward a digital neuromorphic pitch extraction system". In: *IEEE transactions on neural networks* 11.4, pp. 978–987.

Katsiamis, Andreas G, Emmanuel M Drakakis, and Richard F Lyon (2007). "Practical gammatone-like filters for auditory processing". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2007.1, p. 063685.

Kheradpisheh, Saeed Reza, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier (2016). "STDP-based spiking deep neural networks for object recognition". In: *arXiv preprint arXiv:1611.01421*.

Kim, Kyung Hwan, Sung Jin Choi, Jin Ho Kim, and Doo Hee Kim (2009). "An improved speech processing strategy for cochlear implants based on an active nonlinear filterbank model of the biological cochlea". In: *IEEE Transactions on biomedical engineering* 56.3, pp. 828–836.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Kumar, Nagendra, Wolfgang Himmelbauer, Gert Cauwenberghs, and Andreas G Andreou (1998). "An analog VLSI chip with asynchronous interface for auditory feature extraction". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 45.5, pp. 600–606.

Kunkel, Susanne et al. (2017). *NEST 2.12.0*. DOI: 10.5281/zenodo.259534. URL: https://doi.org/10.5281/zenodo.259534.

Lam, MZC, TJ Lee, PY Boey, WF Ng, HW Hey, KY Ho, and PY Cheong (2005). "Factors influencing cardiac auscultation proficiency in physician trainees". In: *Singapore medical journal* 46.1, p. 11.

Lazzaro, John and Carver Mead (1989). "Circuit models of sensory transduction in the cochlea". In: *Analog VLSI Implementation of Neural Systems*. Springer, pp. 85–101.

Lazzaro, John and John Wawrzynek (1995). "A multi-sender asynchronous extension to the AER protocol". In: *Advanced Research in VLSI, 1995. Proceedings., Sixteenth Conference on*. IEEE, pp. 158–169.

Lazzaro, John, John Wawrzynek, Misha Mahowald, Massimo Sivilotti, and Dave Gillespie (1993). "Silicon auditory processors as computer peripherals". In: *Advances in Neural Information Processing Systems*, pp. 820–827.

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989). "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4, pp. 541–551.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, p. 436.

Leong, Monk-Ping, Craig T Jin, and Philip HW Leong (2003). "An FPGA-based electronic cochlea". In: *EURASIP Journal on Applied Signal Processing* 2003, pp. 629–638.

Leung, TS, PR White, WB Collis, E Brown, and AP Salmon (2000). "Classification of heart sounds using time-frequency method and artificial neural networks". In: *Engineering in Medicine and Biology Society, 2000. Proceedings of the 22nd Annual International Conference of the IEEE*. Vol. 2. IEEE, pp. 988–991.

Lichtsteiner, Patrick, Christoph Posch, and Tobi Delbruck (2008). "A 128 $\times$128 120 dB 15μs Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE journal of solid-state circuits* 43.2, pp. 566–576.

Liu, Qian and Steve Furber (2016). "Noisy Softplus: a biology inspired activation function". In: *International Conference on Neural Information Processing*. Springer, pp. 405–412.

Liu, Qian, Yunhua Chen, and Steve Furber (2017). "Noisy Softplus: an activation function that enables SNNs to be trained as ANNs". In: *arXiv preprint arXiv:1706.03609*.

Liu, Shih-Chii, Jörg Kramer, Giacomo Indiveri, Rodney Douglas, et al. (2002). *Analog VLSI: circuits and principles*. MIT press.

Liu, Shih-Chii, André Van Schaik, Bradley A Mincti, and Tobi Delbruck (2010). "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms". In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, pp. 2027–2030.

Liu, Shih-Chii, André van Schaik, Bradley A Minch, and Tobi Delbruck (2014). "Asynchronous Binaural Spatial Audition Sensor With 2x64x4 Channel Output". In: *IEEE transactions on biomedical circuits and systems* 8.4, pp. 453–464.

Liu, Shih-Chii, Tobi Delbruck, Giacomo Indiveri, Adrian Whatley, and Rodney Douglas (2015). *Event-based neuromorphic systems*. John Wiley & Sons.

Liu, Weimin, Andreas G Andreou, and MH Goldstein (1991). "An analog integrated speech front-end based on the auditory periphery". In: *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*. Vol. 2. IEEE, pp. 861–864.

Liu, Weimin, Andreas G Andreou, and Moise H Goldstein (1992). "Voiced-speech representation by an analog silicon model of the auditory periphery". In: *IEEE Transactions on Neural Networks* 3.3, pp. 477–487.

Lyon, Richard (1982). "A computational model of filtering, detection, and compression in the cochlea". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.* Vol. 7. IEEE, pp. 1282–1285.

Lyon, Richard F (2017). *Human and machine hearing*. Cambridge University Press.

Lyon, Richard F and Carver Mead (1988). "An analog electronic cochlea". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36.7, pp. 1119–1134.

Maass, Wolfgang and Christopher M Bishop (2001). *Pulsed neural networks*. MIT press.

Maher, Mary Ann C, Stephen P Deweerth, Misha A Mahowald, and Carver A Mead (1989). "Implementing neural architectures using analog VLSI circuits". In: *IEEE Transactions on circuits and systems* 36.5, pp. 643–652.

Mahowald, Misha (1992). "VLSI analogs of neuronal visual processing: a synthesis of form and function".

Mangione, Salvatore and Linda Z Nieman (1997). "Cardiac auscultatory skills of internal medicine and family practice trainees: a comparison of diagnostic proficiency". In: *Jama* 278.9, pp. 717–722.

Markaki, M, I Germanakis, and Yannis Stylianou (2013). "Automatic classification of systolic heart murmurs". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 1301–1305.

Mathews, Max V, Joan E Miller, F Richard Moore, John R Pierce, and Jean-Claude Risset (1969). *The technology of computer music*. Vol. 969. MIT press Cambridge.

McEwan, Alistair and André van Schaik (2003). "An analogue VLSI implementation of the Meddis inner hair cell model". In: *EURASIP Journal on Applied Signal Processing* 2003, pp. 639–648.

McEwan, Alistair and André van Schaik (2004). "An alternative analog VLSI implementation of the Meddis inner hair cell model". In: *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*. Vol. 4. IEEE, pp. IV–928.

McGuire, Patrick, Jannik Fritsch, Jochen J Steil, F Rothling, Gernot A Fink, Sven Wachsmuth, Gerhard Sagerer, and Helge Ritter (2002). "Multi-modal human-machine communication for instructing robot grasping tasks". In: *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. Vol. 2. IEEE, pp. 1082–1088.

Mead, Carver (1989). *Analog VLSI and Neural Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-05992-4.

Mead, Carver A and Misha A Mahowald (1988). "A silicon model of early visual processing". In: *Neural networks* 1.1, pp. 91–97.

Meddis, Ray (1986). "Simulation of mechanical to neural transduction in the auditory receptor". In: *The Journal of the Acoustical Society of America* 79.3, pp. 702–711.

Meddis, Ray (1988). "Simulation of auditory–neural transduction: Further studies". In: *The Journal of the Acoustical Society of America* 83.3, pp. 1056–1063.

Meddis, Ray, Michael J Hewitt, and Trevor M Shackleton (1990). "Implementation details of a computation model of the inner hair-cell auditory-nerve synapse". In: *The Journal of the Acoustical Society of America* 87.4, pp. 1813–1816.

Miró Amarante, María Lourdes (2013). "Una aportación al procesado neuromórfico de audio basado en modelos pulsantes. Desde la cóclea a la percepción auditiva".

Mookherjee, Soumak, Linda DeBrunner, and Victor DeBrunner (2015). "A low power radix-2 FFT accelerator for FPGA". In: *Signals, Systems and Computers, 2015 49th Asilomar Conference on*. IEEE, pp. 447–451.

Moore, Gordon E (2006). "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp. 114 ff." In: *IEEE solid-state circuits society newsletter* 20.3, pp. 33–35.

Mugliette, Christian, Ivan Grech, Owen Casha, Edward Gatt, and Joseph Micallef (2011). "FPGA active digital cochlea model". In: *Electronics, Circuits and Systems (ICECS), 2011 18th IEEE International Conference on*. IEEE, pp. 699–702.

Nielsen, Andreas Brinch, Lars Kai Hansen, and Ulrik Kjems (2006). "Pitch based sound classification". In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Vol. 3. IEEE, pp. III–III.

Noponen, Anna-Leena, Sakari Lukkarinen, Anna Angerla, and Raimo Sepponen (2007). "Phono-spectrographic analysis of heart murmur in children". In: *BMC pediatrics* 7.1, p. 23.

O'shaughnessy, Douglas (1987). *Speech communication: human and machine*. Universities press.

Patterson, Roy D, K Robinson, J Holdsworth, D McKeown, C Zhang, and M Allerhand (1992). "Complex sounds and auditory images". In: *Auditory physiology and perception*. Elsevier, pp. 429–446.

Patton, Kevin T, Gary A Thibodeau, and Matthew M Douglas (2012). *Essentials of anatomy & physiology*. Elsevier/Mosby.

Pearson, John C, Jack J Gelfand, WE Sullivan, Richard M Peterson, and Clay D Spence (1988). "Neural network approach to sensory fusion". In: *Sensor Fusion*. Vol. 931. International Society for Optics and Photonics, pp. 103–109.

Penrose, Roger and N David Mermin (1990). *The emperor's new mind: Concerning computers, minds, and the laws of physics*.

Perera, Ishanka S, Fathima A Muthalif, Mathuranthagaa Selvarathnam, Madhushanka R Liyanaarachchi, and Nuwan D Nanayakkara (2013). "Automated diagnosis of cardiac abnormalities using heart sounds". In: *Point-of-Care Healthcare Technologies (PHT), 2013 IEEE*. IEEE, pp. 252–255.

Perez-Peña, Fernando, Arturo Morgado-Estevez, Alejandro Linares-Barranco, Angel Jimenez-Fernandez, Francisco Gomez-Rodriguez, Gabriel Jimenez-Moreno, and Juan Lopez-Coronado (2013). "Neuro-inspired spike-based motion: from dynamic vision sensor to robot motor open-loop control through spike-VITE". In: *Sensors* 13.11, pp. 15805–15832.

Perez-Peña, Fernando, J Antonio Leñero-Bardallo, Alejandro Linares-Barranco, and Elisabetta Chicca (2017). "Towards bioinspired close-loop local motor control: A simulated approach supporting neuromorphic implementations". In: *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, pp. 1–4.

Plana, Luis A, Steve B Furber, Steve Temple, Mukaram Khan, Yebin Shi, Jian Wu, and Shufan Yang (2007). "A GALS infrastructure for a massively parallel multiprocessor". In: *IEEE Design & Test of Computers* 24.5.

Plana, Luis A, John Bainbridge, Steve Furber, Sean Salisbury, Yebin Shi, and Jian Wu (2008). "An on-chip and inter-chip communications network for the spinnaker massively-parallel neural net simulator". In: *Proceedings of the second ACM/IEEE International Symposium on Networks-on-Chip*. IEEE Computer Society, pp. 215–216.

Potes, Cristhian, Saman Parvaneh, Asif Rahman, and Bryan Conroy (2016). "Ensemble of feature-based and deep learning-based classifiers for detection of abnormal heart sounds". In: *Computing in Cardiology Conference (CinC), 2016*. IEEE, pp. 621–624.

Rakic, Pasko (1988). "Specification of cerebral cortical areas". In: *Science* 241.4862, pp. 170–176.

Rios-Gutierrez, Fernando, Rocio Alba-Flores, and Spencer Strunic (2012). "Recognition and classification of cardiac murmurs using ANN and segmentation". In: *Electrical Communications and Computers (CONIELECOMP), 2012 22nd International Conference on*. IEEE, pp. 219–223.

Rodríguez Valiente, A, A Trinidad, JR García Berrocal, C Górriz, and R Ramírez Camacho (2014). "Extended high-frequency (9–20 kHz) audiometry reference thresholds in 645 healthy subjects". In: *International journal of audiology* 53.8, pp. 531–545.

Roy, Douglas, Joan Sargeant, Jean Gray, Brian Hoyt, Michael Allen, and Michael Fleming (2002). "Helping family physicians improve their cardiac auscultation skills with an interactive CD-ROM". In: *Journal of Continuing Education in the Health Professions* 22.3, pp. 152–159.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Russakovsky, Olga, Li-Jia Li, and Li Fei-Fei (2015). "Best of both worlds: human-machine collaboration for object annotation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2121–2131.

Schaik, André van, Vincent Chan, and Craig Jin (2009). "Sound localisation with a silicon cochlea pair". In: *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, pp. 2197–2200.

Sen-Bhattacharya, Basabdatta, Sebastian James, Oliver Rhodes, Indar Sugiarto, Andrew Rowley, Alan B Stokes, Kevin Gurney, and Steve B Furber (2018). "Building a Spiking Neural Network Model of the Basal Ganglia on SpiNNaker". In: *IEEE Transactions on Cognitive and Developmental Systems*.

Serrano-Gotarredona, Rafael, Matthias Oster, Patrick Lichtsteiner, Alejandro Linares-Barranco, Rafael Paz-Vicente, Francisco Gómez-Rodríguez, Luis Camuñas-Mesa, Raphael Berner, Manuel Rivas-Pérez, Tobi Delbruck, et al. (2009). "CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory–processing–learning–actuating system for high-speed visual object

recognition and tracking". In: *IEEE Transactions on Neural Networks* 20.9, pp. 1417–1438.

Serrano-Gotarredona, Teresa and Bernabé Linares-Barranco (2013). "A 128 $\times$ 128 1.5% Contrast Sensitivity 0.9% FPN 3 μs Latency 4 mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Preamplifiers". In: *IEEE Journal of Solid-State Circuits* 48.3, pp. 827–838.

Shadlen, Michael N and William T Newsome (1994). "Noise, neural codes and cortical organization". In: *Current opinion in neurobiology* 4.4, pp. 569–579.

Shepherd, Gordon M (2003). *The synaptic organization of the brain*. Oxford University Press.

Shiraishi, Hisako (2003). "Design of an analog VLSI cochlea".

Shore, SE (2009). *Auditory/somatosensory interactions*. Elsevier.

Singh, Mandeep and Amandeep Cheema (2013). "Heart sounds classification using feature extraction of phonocardiography signal". In: *International Journal of Computer Applications* 77.4.

Sivilotti, Massimo Antonio (1991). *Wiring considerations in analog VLSI systems, with application to field-programmable networks.*

Skottun, Bernt C, Trevor M Shackleton, Robert H Arnott, and Alan R Palmer (2001). "The ability of inferior colliculus neurons to signal differences in interaural delay". In: *Proceedings of the National Academy of Sciences* 98.24, pp. 14050–14054.

Slaney, Malcolm et al. (1993). "An efficient implementation of the Patterson-Holdsworth auditory filter bank". In: *Apple Computer, Perception Group, Tech. Rep* 35.8.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Stevens, Stanley Smith, John Volkmann, and Edwin B Newman (1937). "A scale for the measurement of the psychological magnitude pitch". In: *The Journal of the Acoustical Society of America* 8.3, pp. 185–190.

Stromatias, Evangelos, Daniel Neil, Michael Pfeiffer, Francesco Galluppi, Steve B Furber, and Shih-Chii Liu (2015). "Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms". In: *Frontiers in neuroscience* 9, p. 222.

Strunic, Spencer L, Fernando Rios-Gutiérrez, Rocío Alba-Flores, Glenn Nordehn, and S Bums (2007). "Detection and classification of cardiac murmurs using segmentation techniques and artificial neural networks". In: *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*. IEEE, pp. 397–404.

Summerfield, Clive D and Richard F Lyon (1992). "ASIC implementation of the Lyon cochlea model". In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. Vol. 5. IEEE, pp. 673–676.

Thakur, Chetan Singh, Tara Julia Hamilton, Jonathan Tapson, André van Schaik, and Richard F Lyon (2014). "FPGA Implementation of the CAR Model of the Cochlea". In: *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, pp. 1853–1856.

Thorpe, Simon J, Adrien Brilhault, and José-Antonio Perez-Carrasco (2010). "Suggestions for a biologically inspired spiking retina using order-based coding". In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, pp. 265–268.

Ursino, Mauro, Cristiano Cuppini, and Elisa Magosso (2016). "Sensory fusion: A neurocomputational approach". In: *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on*. IEEE, pp. 1–6.

Van Schaik, André and Eric Fragnière (2001). "Pseudo-voltage domain implementation of a 2-dimensional silicon cochlea". In: *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*. Vol. 3. IEEE, pp. 185–188.

Van Schaik, André, Eric Fragnière, and Eric A Vittoz (1996). "Improved silicon cochlea using compatible lateral bipolar transistors". In: *Advances in neural information processing systems*, pp. 671–677.

Watts, Lloyd, Douglas A Kerns, Richard F Lyon, and Carver A Mead (1992). "Improved implementation of the silicon cochlea". In: *IEEE Journal of Solid-state circuits* 27.5, pp. 692–700.

Wen, Bo and Kwabena Boahen (2009). "A silicon cochlea with active coupling". In: *IEEE transactions on biomedical circuits and systems* 3.6, pp. 444–455.

Westerman, Wayne C, David PM Northmore, and John G Elias (1997). "Neuromorphic synapses for artificial dendrites". In: *Analog Integrated Circuits and Signal Processing* 13.1-2, pp. 167–184.

Williams, Gethin and Steve Renals (1997). "Confidence measures for hybrid HMM/ANN speech recognition." In: International Speech Communication Association.

World Health Organization (2017). *Cardiovascular Disease*. http://www.who.int/cardiovascular_diseases.

Xu, Ying, Chetan S Thakur, Ram K Singh, Tara Julia Hamilton, Runchun M Wang, and André van Schaik (2018). "A FPGA implementation of the CAR-FAC cochlear model". In: *Frontiers in neuroscience* 12, p. 198.

Yang, Minhao, Chen-Han Chien, Tobi Delbruck, and Shih-Chii Liu (2016). "A 0.5 V 55μW 64$\times$2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing". In: *IEEE Journal of Solid-State Circuits* 51.11, pp. 2554–2569.

Yu, Theodore, Andrew Schwartz, John Harris, Malcolm Slaney, and Shih-Chii Liu (2009). "Periodicity detection and localization using spike timing from the AER EAR". In: *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. IEEE, pp. 109–112.

Zabihi, Morteza, Ali Bahrami Rad, Serkan Kiranyaz, Moncef Gabbouj, and Aggelos K Katsaggelos (2016). "Heart sound anomaly and quality detection

using ensemble of neural networks without segmentation". In: *Computing in Cardiology Conference (CinC), 2016*. IEEE, pp. 613–616.

Zwicker, Eberhard (1961). "Subdivision of the audible frequency range into critical bands (Frequenzgruppen)". In: *The Journal of the Acoustical Society of America* 33.2, pp. 248–248.

Zwicker, Eberhard and Hugo Fastl (2013). *Psychoacoustics: Facts and models*. Vol. 22. Springer Science & Business Media.

Part II

Set of papers

Appendix A

NAVIS: Neuromorphic Auditory VISualizer

Authors

- Juan Pedro Dominguez-Morales
- Angel Jimenez-Fernandez
- Manuel J. Dominguez-Morales
- Gabriel Jimenez-Moreno

Publication

Title: Neurocomputing

Type: Journal Article

Issue: 237

Publisher: Elsevier

Date: May 2017

Pages: 418-422

ISSN: 0925-2312

Neurocomputing 237 (2017) 418–422

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Original software publication

NAVIS: Neuromorphic Auditory VISualizer Tool

Juan P. Dominguez-Morales*, A. Jimenez-Fernandez, M. Dominguez-Morales, G. Jimenez-Moreno

Robotic and Technology of Computers Lab. from the University of Seville, Spain

ARTICLE INFO

Keywords:
Neuromorphic engineering
Spiking neural networks
Address-Event-Representation
Auditory sensor

ABSTRACT

This software presents diverse utilities to develop the first post-processing layer using the neuromorphic auditory sensor's (NAS) information. The NAS used implements a cascade filters architecture in FPGA, imitating the behavior of the basilar membrane and inner hair cells, working with the sound information decomposed into its frequency components as spike streams. The neuromorphic hardware interface Address-Event-Representation (AER) is used to propagate auditory information out of the NAS, emulating the auditory vestibular nerve. Using the packetized information (aedat files) generated with jAER software plus an AER to USB computer interface, NAVIS implements a set of graphs that allows to represent the auditory information as cochleograms, histograms, sonograms, etc. It can also split the auditory information into different sets depending on the activity level of the spike streams. The main contribution of this software tool is its capability to apply complex audio post-processing treatments and representations, which is a novelty for spike-based systems in the neuromorphic community. This software will help neuromorphic engineers to build sets for the training of spiking neural networks (SNN).

Software metadata

(Executable) Software metadata description

Current software version	*1.0.0.2.*
Permanent link to executables of this version	https://github.com/Neurocomputing/NEUCOM-D-16-00057
Legal Software License	*GNU General Public License (GPL)*
Computing platform / Operating System	*Microsoft Windows*
Installation requirements & dependencies	*Microsoft .NET Framework 4.5 or greater*
If available Link to user manual – if formally published include a reference to the publication in the reference list	https://github.com/jpdominguez/NAVIS-Tool/blob/master/NAVIS/NAVIS_LatestBuild/UserManual.pdf
Support email for questions	jpdominguez@atc.us.es

* Correspondence to: Department of Architecture and Technology of Computers, University of Seville, Av. Reina Mercedes s/n. 41012, Sevilla, Spain.
E-mail address: jpdominguez@atc.us.es (J.P. Dominguez-Morales).

http://dx.doi.org/10.1016/j.neucom.2016.12.046
Received 18 January 2016; Received in revised form 18 November 2016; Accepted 17 December 2016
Available online 23 December 2016
0925-2312/

Code metadata

Code metadata description	
Current Code version	*1.0.0.2.*
Permanent link to code / repository used of this code version	https://github.com/Neurocomputing/NEUCOM-D-16-00057
Legal Code License	*GNU General Public License (GPL)*
Code Versioning system used	*Git*
Software Code Language used	*C Sharp (C#)*
Compilation requirements, Operating environments & dependencies	*Microsoft Visual Studio, Microsoft .NET Framework 4.5 or greater*
If available Link to developer documentation / manual	https://github.com/jpdominguez/NAVIS-Tool/blob/master/NAVIS/NAVIS_VisualStudioProject/SoftwareArchitecture.pdf https://github.com/jpdominguez/NAVIS-Tool/blob/master/NAVIS/NAVIS_VisualStudioProject/NAVIS.XML
Support email for questions	jpdominguez@atc.us.es

1. Introduction

Neuromorphic engineering is a discipline that studies, designs and implements hardware and software that mimic the way in which nervous systems work, focusing its main inspiration in how the brain solves complex problems easily. Currently, the neuromorphic community has a set of neuromorphic hardware, such as sensors [1], learning circuits [2], neuromorphic information filters and feature extractors [3], and robotic and motor controllers [4]. In the field of neuromorphic sensors, diverse neuromorphic cochleae can be found [5]. These sensors are able to decompose audio signals into frequency bands, represent them as streams of short pulses, called spikes, using the Address-Event Representation [6], and then make them interface with other neuromorphic layers. On the other hand, there are several software tools in the community, like NENGO [7] or BRIAN [8], for spiking neural networks simulation with or without learning; or jAER [9], for real-time visualization and software processing of AER streams captured from the hardware using specified interfaces [10]. The aim of the software presented in this paper, called NAVIS, is to help the neuromorphic community to work with cochleae data in order to visualize and adapt this information to build training sets for later learning. To demonstrate these software functionalities, a 64-channel binaural Neuromorphic Auditory Sensor (NAS) for FPGA [5] has been used together with an USB-AER interface [10]. NAS responses are stored as aedat files through jAER. However, since this software works with aedat files, it can work with any cochleae sensor connected to jAER using its aedat files.

NAVIS provides neuromorphic engineers with a set of functions: (1) detailed data visualization through cochleograms, sonograms, histograms, average activity and channels disparity; (2) it performs a set of algorithms for different stream splits; (3) it can remove silences and generate a set of new files with the most relevant auditory information; and (4) it can be a test bed for building data sets for the training of spiking neural networks. For example, it can reproduce an audio file, with all the cases and samples to train a Spiking Neural Network (SNN), i.e. SpiNNaker [11], and it can store this information into aedat files, which can be used in jAER. After recording an aedat file, NAVIS is used to visualize features and measure the quality of the auditory information. It can search silences between samples and use them to split the original file into several streams with short sentences or words; and, finally, it can store a new set of aedat files without silences, providing one file per sample.

Other software applications that allow the post-processing of audio information are, as an example of sample-based or event-based, Audacity [12] and jAER [9], respectively. Audacity is a free, open source digital audio editor that provides a set of tools for post-processing information. These tools are mostly the same as the ones implemented by NAVIS: sonogram, histogram, automatic split, manual split, average activity and disparity between left and right channels; however, Audacity is only capable of working with discrete audio samples, instead of spiking information obtained from a neuromorphic sensor (as jAER and NAVIS do). jAER only allows to represent the cochleogram for spiking audio information, lacking in several functionalities that are necessary when post-processing this kind of information. NAVIS provides functionalities that are similar to those in Audacity, but using spike-coded audio information to aid neuromorphic engineers, which is a novelty in the neuromorphic community.

2. Software framework

The Neuromorphic Auditory VISualizer (NAVIS) Tool is able to read aedat files and perform a set of algorithms of the binary data contained in order to obtain results and graphical representations that will provide relevant information about the events communicated during a time period. Once the file is loaded, it gives a set of functions through the menus and the left side tool bar, which can be applied to the data. The stream of events contained in the aedat file used in the examples corresponds to a young woman reading the first sentence of the famous Spanish novel *The Ingenious Gentleman Don Quixote of La Mancha*: "*En un lugar de La Mancha*". Fig. 1 shows the algorithms that have been implemented for each of the main NAVIS functionalities, whose description and result are detailed in the next subsections.

2.1. Cochleogram

After choosing the aedat file to be loaded, the internal data decodification process and the subsequent cochleogram calculation and representation in the main application window are launched. The main window of the application and the output of the cochleogram can be seen in Fig. 2, where the X axis represents time (μs) and the Y axis is the AER address assigned to each frequency band. Each dot corresponds to an event that has been fired in a particular AER address at a specific time. The picture shows the AER events fired for both cochleae: the left one at the bottom and the right one on top, embracing up to 256 addresses for the two sets of 64 channels of the binaural NAS [5], where each channel has positive and negative events. Lower addresses belong to the events of higher frequency channels. The response of the events appears to be "lying", this is because of the cascade architecture of the internal filters (Spike Low Pass Filter –SLPF– [3]), since each filter induces a small phase increment, which increases as the spikes go across a SLPF [5].

2.2. Sonogram

Fig. 3 represents the spike rate of both cochleae in a color map, where X axis is time, Y axis is the NAS channel, and the color is the relative spike rate of the channel for a particular time period, software settable. Each one of the words contained in the sentence "*En un lugar de La Mancha*" can be clearly distinguished.

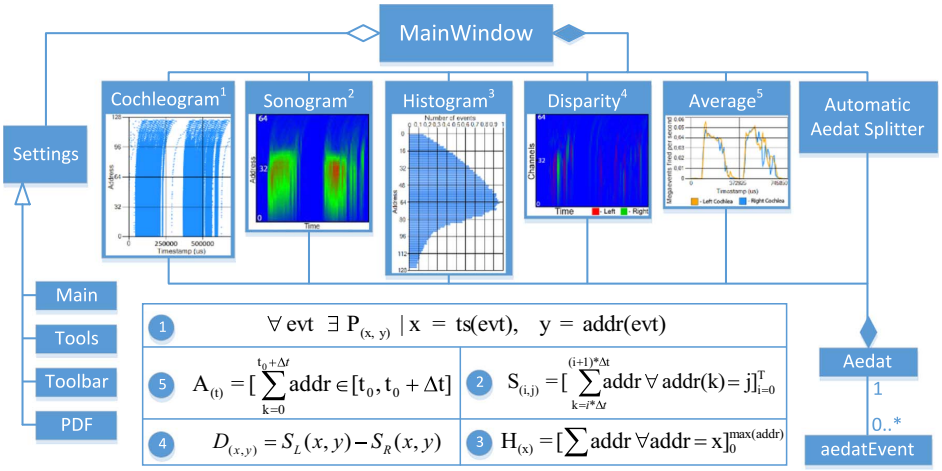


Fig. 1. Block diagram of the overall software architecture describing the algorithms used in the main NAVIS' functionalities.

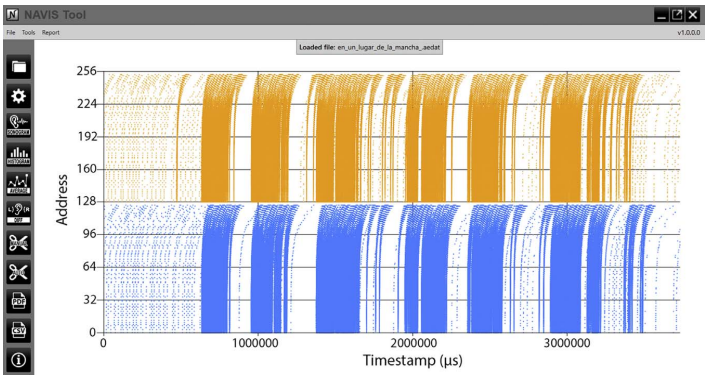


Fig. 2. Cochleogram representing "En un lugar de La Mancha". Left 64-channels represented in blue and right ones in orange.(For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

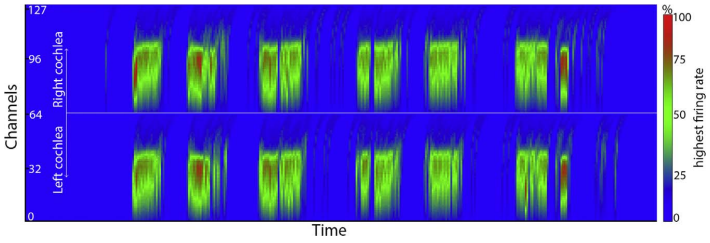


Fig. 3. Sonogram. Left 64-channels represented in bottom and right ones in top.

2.3. Histogram

For certain applications it is important to know which cochlea channels are the ones that fire more events. The histogram is the appropriate chart to measure this information. Fig. 4 shows the resulting histogram for the same test scenario. The X axis represents the 256 possible AER address values, while the Y axis is the number of events fired. An option is included for the user to choose the histogram to be normalized or not, as shown in Fig. 4.

2.4. Disparity between left and right channels

The tool allows to know the difference between both cochleae (searching for the predominant one in a period of time), which is very

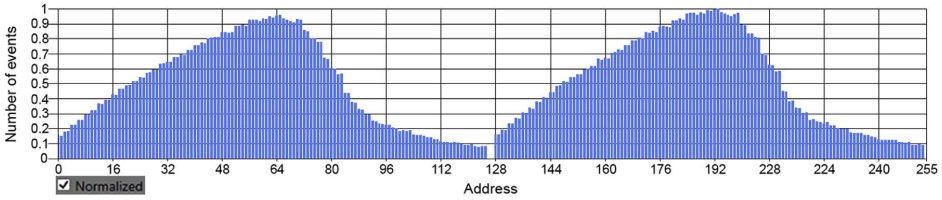


Fig. 4. Histogram. Left 64-channels represented in the left and right ones in the right.

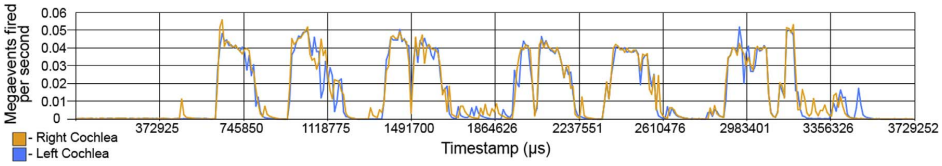


Fig. 5. Average activity of both NAS.

useful for source echolocation: after it, dominant events polarity and spike rate can determine the orientation and distance of the sound source (after a calibration or learning step).

2.5. Average activity of both channels

It is really important to know the activity of the NAS in terms of the number of AER events fired per second in a particular time period, either for source echolocation as the tool presented above or to quantify the firing rate of both NAS. This tool generates a 2-axis chart (see Fig. 5), where the X axis represents time (timestamp, µs) and the Y axis is the number of mega-events (10^6 events) fired per second, calculated by counting the number of events produced between a time period (integration period) and dividing it by this number.

2.6. Aedat files split

One of the main aims of NAVIS is to split aedat files automatically into different files depending on the fire rate of both channels. This analysis is performed over a time period. A division is stablished when there is not enough activity to consider a sound, while removing silences. These sections are saved as single aedat files. The main goal of this utility is to build training sets for SNNs: after recording into one file a sequence of audio samples, it is able to distinguish all individual stimuli and store them separately.

NAVIS allows to manually extract a specific section of the original file and save it separately. It also implements a splitting function based on the channels' average activity. Two configurable parameters are needed for this purpose: (1) Noise Threshold (% of the number of events that an integration period needs in order to be considered as a sound instead of a noise); and (2) Noise Tolerance (consecutive integration periods that are needed to be detected above the Noise Threshold in order to be considered as a real sound instead of a noise). When a period with not enough rate is detected, a new split is created if the process passed through *Noise Tolerance* consecutive times (or more) detecting sound; otherwise, this information will be taken as noise and will be omitted in the results of this function.

To test the performance of the Automatic Aedat Splitter tool, a new experiment was carried out. The experiment consisted of automatically splitting an aedat file that contained eight different pure tones (different frequencies, 1 s duration each) with a second of silence between them, and then calculating the average error by subtracting the duration of each split from the length that they should have if the pure tone was perfectly extracted from the audio sample. Different integration period values were used and the results can be seen in Fig. 6.

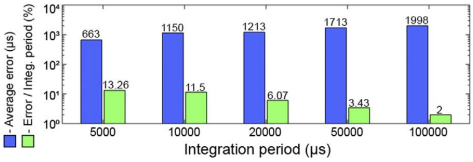


Fig. 6. Average error (µs) and error/integration period percentage obtained when automatically splitting an aedat file that contains 8 one-second duration pure tones using different integration period values.

3. Performance analysis

The information of each AER event is stored in an object with three attributes: address, timestamp and ID (position within the file). The loading process adds these objects to a list, which will contain the entire information of the events stored in the aedat file (more information about the software architecture and data model can be found in the NAVIS wiki [13]). As large data size files (more than a million of events in this example) are frequently used in this application, software optimization techniques need to be applied in order to work with the information in a fast way and reduce the CPU's load as much as possible. Therefore, LINQ (Language-Integrated Query) [14] query expressions were used. LINQ is a Microsoft .NET platform component that offers an API called Standard Query Operator (SQO), which allows to handle large lists (for example aedat files) in the most optimized way.

Table 1 shows execution times for the functions that have been implemented in the NAVIS Tool, measured with Microsoft Visual Studio Profiling Tools. For this study, three different processors were used, so that the scalability of the operations could be analyzed. In addition to the total and local time of each function, the table also presents the percentage of the total time and the time per event (TPE) for every tool. This last number is extremely important due to the fact that the execution time directly depends on the total number of events that the file has. We can observe that generating the pdf report is the function that takes the longest time to execute (around 45% of the total), since the process to create that file needs to calculate the output from the rest of the functions.

4. Conclusions

In this manuscript we have presented the design and development details of a software application that is able to load AER streams stored in aedat files obtained from a 64-channel binaural NAS, captured using

Table 1
Comparison on execution times per function between three different processors.

Functions	Processors					
	Celeron P4500 (1.87 GHz)		i5-4460 (3.2 GHz)		i7-4770 (3.5 GHz)	
	Time (ms, %)	TPE (ms)	Time (ms, %)	TPE (ms)	Time (ms, %)	TPE (ms)
Load Aedat	3346.76, 5.293%	$30.96*10^{-4}$	1266.99, 5.556%	$11.72*10^{-4}$	1374.54, 6.346%	$12.70*10^{-4}$
Sonogram	8883.82, 14.05%	$82.21*10^{-4}$	3329.38, 14.60%	$30.80*10^{-4}$	3001.86, 13.859%	$27.76*10^{-4}$
Histogram	70.82, 0.112%	$0.657*10^{-4}$	28.96, 0.127%	$0.268*10^{-4}$	35.96, 0.166%	$0.33*10^{-4}$
Disparity	8668.83, 13.71%	$80.11*10^{-4}$	3117.30, 13.67%	$28.83*10^{-4}$	2962.22, 13.676%	$27.39*10^{-4}$
Average activity	4099.83, 6.484%	$37.92*10^{-4}$	1287.97, 5.648%	$11.91*10^{-4}$	1287.04, 5.942%	$11.90*10^{-4}$
Automatic split	10040.92, 15.88%	$92.87*10^{-4}$	3557.42, 15.60%	$32.91*10^{-4}$	3290.15, 15.19%	$30.43*10^{-4}$
Manual split	123.93, 0.196%	$0.114*10^{-4}$	54.05, 0.237%	$0.499*10^{-4}$	114.36, 0.528%	$1.05*10^{-4}$
Generate PDF	27998.24, 44.28%	$258.9*10^{-4}$	10161.46, 44.56%	$93.98*10^{-4}$	9593.86, 44.293%	$88.74*10^{-4}$
Total time	63230, 100%	$584*10^{-4}$	22804, 100%	$211*10^{-4}$	21660, 100%	$200.3*10^{-4}$

jAER software and USB-AER hardware interface. A data model has been designed to store and use the information extracted from aedat files. This model represents the information efficiently and facilitates its subsequent access and processing. A set of functionalities have been designed to apply operations to the data gathered from the aedat file and to generate graph results. To improve execution times and reduce resources used in memory, these functions have been optimized using LINQ query expressions and lambda functions.

Finally, the tool includes a functionality that is able to split an aedat file into several ones, both manually and automatically, depending on the average activity of the cochlea. This is a very useful capability for creating training sets for spiking neural networks. The software tool presented can help neuromorphic researchers to process and evaluate large amounts of cochlear information in an easy and efficient way.

Acknowledgements

We want to thank Alejandro Linares-Barranco for his support and contribution. This work is supported by the Spanish government grant (with support from the European Regional Development Fund) COFNET (TEC2016-77785-P).

References

[1] P. Lichtsteiner, et al., A 128×128 120dB 15μs latency asynchronous temporal contrast vision sensor,", IEEE J. Solid-State Circuits 43 (2008) 566–576.
[2] R. Serrano-Gotarredona, et al., On real-time AER 2-D convolutions hardware for neuromorphic spike-based cortical processing, IEEE Trans. Neural Netw. 19 (2008) 1196–1219.
[3] A. Jiménez-Fernández, et al., Building blocks for spikes signal processing, IJCNN (2010).
[4] A. Jiménez-Fernández, et al., A neuro-inspired spike-based PID motor controller for multi-motor robots with low cost FPGAs, Sensors 12 (2012) 3831–3856.
[5] A. Jimenez-Fernandez, et al., A binaural Neuromorphic auditory sensor for FPGA: a spike signal processing approach (vol. PP, Issue 99)IEEE Trans. Neural Netw. (2016) (vol. PP, Issue 99).
[6] K. Boahen, Point-to-point connectivity between neuromorphic chips using address events, IEEE Trans. Circuits Syst. II Analog Digit. Signal Process. 47 (2000) 416–434.
[7] T. Bekolay, et al., Nengo: a Python tool for building large-scale functional brain models.,, Front. Neuroinform (2014).
[8] D. Goodman, R. Brette, Brian: a simulator for spiking neural networks in Python.,, Front. Neuroinform. (2008).
[9] T.Delbrück, jAER Open Source Project,2007. [Online]. Available: ⟨http://sourceforge.net/p/jaer/wiki/Home/⟩.
[10] R. Berner, et al., A 5 Meps $100 USB20.0 address-event monitor-Sequencer interface,, IEEE ISCAS (2007).
[11] E. Painkras, et al., SpiNNaker: a 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation,", IEEE J. Solid-State Circuits 48 (8) (2013) 1943–1953.
[12] "Audacity."⟨http://www.audacityteam.org⟩.
[13] J.Dominguez-Morales, NAVIS Tool Wiki.⟨https://github.com/jpdominguez/NAVIS-Tool/wiki/Software-architecture⟩.
[14] LINQ:.NETLanguage-Integrated Query.⟨https://msdn.microsoft.com/en-us/library/bb308959.aspx⟩.

Juan P. Dominguez-Morales received the B.S. degree in computer engineering in 2014 and the M.S. degree in computer engineering and networks in 2015 from the University of Seville, Sevilla, Spain. Since October 2015, he has been a Ph.D. student in the Computer Architecture and Technology Department, at University of Seville. His research interests include neuromorphic engineering, spiking neural networks and neuromorphic sensors.

Angel Jimenez-Fernandez received the B.S. degree in computer engineering in 2005, the M.S. degree in industrial computer science in 2007 and the Ph.D. in Neuromorphic Engineering in 2010 from the University of Seville, Sevilla, Spain. Since October 2007, he has been an Assistant Professor of Computer Architecture and Technology at the University of Seville. In April 2011, he was promoted to Associate Professor. His research interests include neuromorphic engineering applied to robotics, real-time spikes signal processing, neuromorphic sensors, field programmable gate array (FPGA) digital design, embedded systems development, high-speed serial communications, and smart sensors networking.

Manuel Domínguez-Morales received the B.S. degree in Computer Science from the University of Seville (Spain) in 2008. He obtained a M.S. degree in Software Engineering in 2009 and a M.S. degree in Computer Engineering in 2014, both from the University of Seville (Spain). Finally, he obtained the Ph.D. on Industrial Informatics in 2014 from the same university. From January 2009 to June 2014 he has worked as Assistant Professor at the Computer Architecture and Technology Department of the University of Seville. Since July 2014 he works as Postdoc Researcher and Lecturer in the same department. His research fields are focused on computer vision, image processing, signal processing, programmable hardware digital design, computer architecture, robotics and e-Health. He has more than 30 publications on international journals and congresses.

Gabriel Jiménez-Moreno received the M.S. degree in physics (electronics) and the Ph.D. degree from the University of Seville, Seville, Spain, in 1987 and 1992, respectively. After working with Alcatel, he was granted a Fellowship from the Spanish Science and Technology Commission (CICYT). Currently, he is an Associate Professor of computer architecture at the University of Seville. From 1996 until 1998, he was Vice-Dean of the E.T.S. Ingenieria Informatica, University of Seville. He participated in the creation of the Department of Computer Architecture (also at University of Seville) and since 2013, has been its Director. He is the author of various papers and research reports on robotics, rehabilitation technology, and computer architecture. He has directed three national research projects on neuromorphic systems. His research interests include neural networks, vision processing systems, embedded systems, computer interfaces, and computer architectures.

Appendix B

Multilayer Spiking Neural Network for Audio Samples Classification Using SpiNNaker

Authors

- Juan Pedro Dominguez-Morales
- Angel Jimenez-Fernandez
- Antonio Rios-Navarro
- Elena Cerezuela-Escudero
- Daniel Gutierrez-Galan
- Manuel J. Dominguez-Morales
- Gabriel Jimenez-Moreno

Publication

Title: Artificial Neural Networks and Machine Learning

Type: Conference Paper

Conference Name International Conference on Artificial Neural Networks (ICANN 2016)

Place Barcelona, Spain. **Date:** September 2016

Publisher: Springer

Pages: 45-53

ISSN: 0302-9743. **ISBN:** 978-3-319-44777-3

Multilayer Spiking Neural Network for Audio Samples Classification Using SpiNNaker

Juan Pedro Dominguez-Morales(✉), Angel Jimenez-Fernandez, Antonio Rios-Navarro, Elena Cerezuela-Escudero, Daniel Gutierrez-Galan, Manuel J. Dominguez-Morales, and Gabriel Jimenez-Moreno

Robotic and Technology of Computers Lab,
Department of Architecture and Technology of Computers,
University of Seville, Seville, Spain
{jpdominguez,ajimenez,arios,ecerezuela,
dgutierrez,mdominguez,gaji}@atc.us.es
http://www.atc.us.es

Abstract. Audio classification has always been an interesting subject of research inside the neuromorphic engineering field. Tools like Nengo or Brian, and hardware platforms like the SpiNNaker board are rapidly increasing in popularity in the neuromorphic community due to the ease of modelling spiking neural networks with them. In this manuscript a multilayer spiking neural network for audio samples classification using SpiNNaker is presented. The network consists of different leaky integrate-and-fire neuron layers. The connections between them are trained using novel firing rate based algorithms and tested using sets of pure tones with frequencies that range from 130.813 to 1396.91 Hz. The hit rate percentage values are obtained after adding a random noise signal to the original pure tone signal. The results show very good classification results (above 85 % hit rate) for each class when the Signal-to-noise ratio is above 3 decibels, validating the robustness of the network configuration and the training step.

Keywords: SpiNNaker · Spiking neural network · Audio samples classification · Spikes · Neuromorphic auditory sensor · Address-Event Representation

1 Introduction

Neuromorphic engineering is a discipline that studies, designs and implements hardware and software with the aim of mimicking the way in which nervous systems work, focusing its main inspiration on how the brain solves complex problems easily. Nowadays, the neuromorphic community has a set of neuromorphic hardware tools available such as sensors [1, 2], learning circuits [3, 4], neuromorphic information filters and feature extractors [5, 6], robotic and motor controllers [7, 8]. In the field of neuromorphic sensors, diverse neuromorphic cochleae can be found [2, 9, 10]. These sensors are able to decompose the audio in frequency bands, and represent them as streams of short pulses, called spikes, using the Address-Event Representation (AER) [11] to interface with other neuromorphic layers. On the other hand, there are several software tools in the community for spiking neural networks (SNN) simulation, i.e. NENGO [12] and

A.E.P. Villa et al. (Eds.): ICANN 2016, Part I, LNCS 9886, pp. 45–53, 2016.
DOI: 10.1007/978-3-319-44778-0_6

BRIAN [13]; or jAER [14] for real-time visualization and software processing of AER streams captured from the hardware using specific interfaces [15]. Hardware platforms like the SpiNNaker board [16] allows to develop and implement complex SNN easily using a high-level programming language such as Python and the PyNN [17] library.

This manuscript presents a novel multilayer SNN architecture built in SpiNNaker which has been trained for audio samples classification using a firing rate based algorithm. To test the network behavior and robustness, a 64-channel binaural Neuromorphic Auditory Sensor (NAS) for FPGA [10] has been used together with an USB-AER interface [15] (Fig. 1) and the jAER software, allowing to produce different pure tones with frequencies varying from 130.813 Hz to 1396.91 Hz, record the NAS response storing the information in aedat files through jAER and use these files as input for the SNN that has been implemented in the SpiNNaker board.

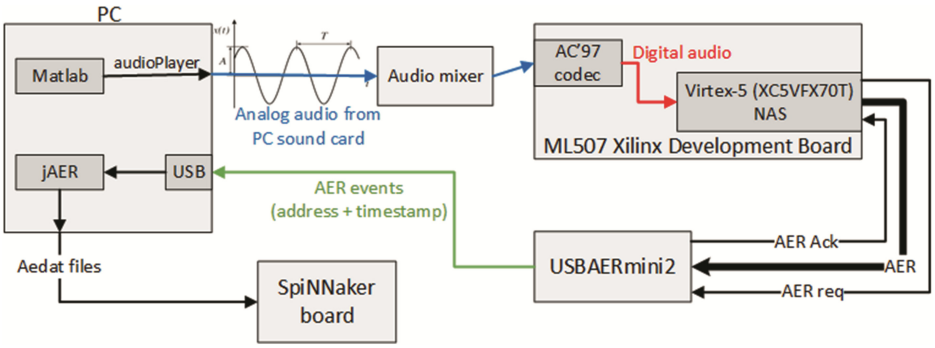


Fig. 1. Block diagram of the system

The paper is structured as follows: Sect. 2 presents the number of neurons, layers and connections of the SNN. Then, Sect. 3 describes the training algorithm used in every layer for the audio samples classification. Section 4 describes the test scenario, including information about the input files. Then, Sect. 5 presents the experimental results of the audio samples classification when using the inputs described in Sect. 4. Finally, Sect. 6 presents the conclusions of this work.

2 Hardware Setup

The standalone hardware used in this work consists of two main parts: the 64-channel NAS connected to the USB-AER interface for generating a spike stream for each audio sample, and the SpiNNaker for back-end computation and deployment of the SNN classifier.

2.1 Neuromorphic Auditory Sensor (NAS)

A Neuromorphic Auditory Sensor (NAS) is used as the input layer of our system. This sensor converts the incoming sound into a train of rate-coded spikes and processes them

using Spike Signal Processing (SSP) techniques for FPGA [5]. NAS is composed of a set of Spike Low-pass Filters (SLPF) implementing a cascade topology, where SLPF's correlative spike outputs are subtracted, performing a bank of equivalent Spikes Band-pass Filters (SBPF), and decomposing input audio spikes into spectral activity [10]. Finally, SBPF spikes are collected using an AER monitor, codifying each spike using the Address-Event Representation, and propagating AER events through a 16-bit parallel asynchronous AER port [11].

NAS designing is very flexible and fully customizable, allowing neuromorphic engineers to build application-specific NASs, with diverse features and number of channels. In this case, we have used a 64-channel binaural NAS, with a frequency response between 20 Hz and 22 kHz, and a dynamic range of +75 dB, synthesized for a Virtex-5 FPGA. Figure 1 shows a NAS implemented in a Xilinx development board, and a USB-AER mini2 board, that implements a bridge between AER systems and jAER in a PC (Fig. 2).

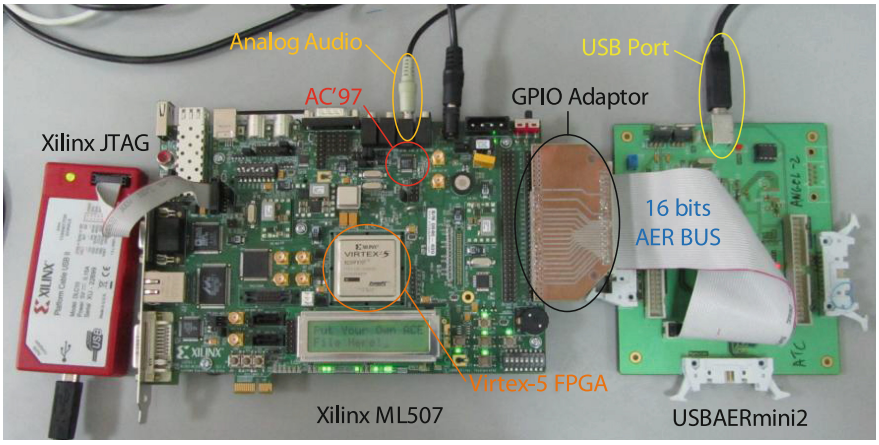


Fig. 2. 64-channel binaural NAS implemented in a Xilinx ML507 FPGA connected to an USB-AER mini2 board.

2.2 Spiking Neural Network Architecture (SpiNNaker)

SpiNNaker is a massively-parallel multi-core computing system designed for modelling very large spiking neural networks in real time. Each SpiNNaker chip comprises 18 general-purpose ARM968 cores, running at 200 MHz, communicating via packets carried by a custom interconnect fabric. Packets are transmitted and their transmission is brokered entirely by hardware, giving the overall engine an extremely high bisection bandwidth. The Advanced Processor Technologies Research Group (APT) [18] in Manchester are responsible for the system architecture and the design of the SpiNNaker chip itself.

In this work, a SpiNNaker 102 machine was used. The 102 machine, Fig. 3, is a 4-node circuit board and hence has 72 ARM processor cores, which are typically deployed as 64 application cores, 4 Monitor Processors and 4 spare cores. The 102 machine

requires a 5 V 1 A supply, and can be powered from some USB2 ports. The control and I/O interface is a single 100 Mbps Ethernet connection.

Fig. 3. SpiNNaker 102 machine.

3 Leaky Integrate-and-Fire Spiking Neural Network

The SpiNNaker platform allows to implement a specific spiking neuron model and use it in any SNN deployed on the board thanks to the PyNN package. Leaky Integrate-and-Fire (LIF) neurons have been used in a 3-layer SNN architecture for audio samples classification.

- **Input layer.** This layer receives the stream of AER events fired for the audio samples captured as aedat files through jAER. The number of input neurons is equal to the number of channels that the NAS has. As a 64-channel NAS (64 different AER addresses) was used in this work, the input layer consists of 64 LIF neurons.
- **Hidden layer.** The hidden layer has the same number of neurons as the desired number of classes to be classified in the output layer. As an example, this layer should consist of eight LIF neurons if eight different audio samples are expected to be classified.
- **Output layer.** As the previous layer, this also has as many neurons as output classes. The firing output of the neurons in this layer will determine the result of the classification.

Figure 4 shows the SNN architecture. Connections between layers are achieved using the FromListConnector method from PyNN, meaning that the source, destination and weight of the connection are specified manually. Using other connectors from this package will result on having the same weight in all the connections between consecutive layers, instead of a different value for each. In this architecture, each neuron in a layer is connected to every neuron in the next layer, and the weight value is obtained from the training step, which is described in Sect. 4. The threshold voltage of the neurons in the hidden layer is 15 mV, while this voltage is 10 mV in the neurons in the output layer.

Decay rate and refractory period are the same for both layers: 20 mV/ms and 2 ms, respectively.

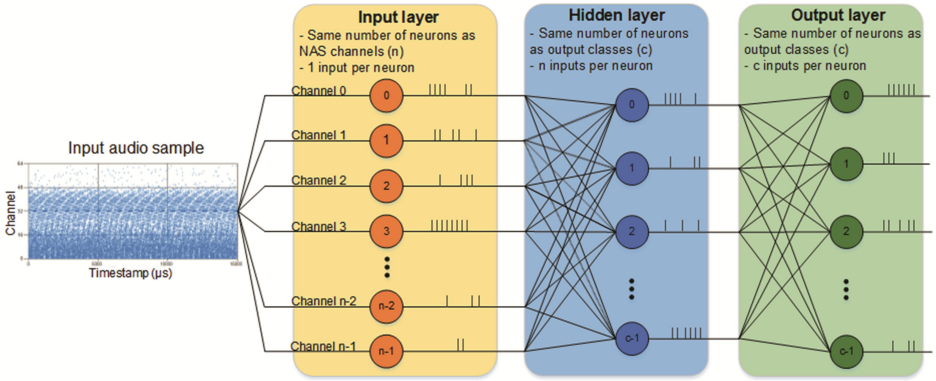


Fig. 4. SNN architecture using an audio sample aedat file as input.

4 Training Phase for Audio Classification

In the previous section, each of the three layers comprising the network were described. The training phase is performed offline and supervised. The main objective of this training is to obtain the weight values of the connections between the input and the hidden layer and between the hidden and the output layers for further audio samples classification. Therefore, two different training steps need to be done.

The weights of the first step of the training phase are obtained from the normalized spike firing activity for each NAS channel using a set of audio samples similar to those to be recognized (same amplitude, duration and frequencies). The firing rate for a specific channel ($FR_{channel\_i}$) is obtained by dividing the number of events produced in that channel by the NAS firing rate (FR_T), which is the number of events fired in the NAS in a particular time period.

$$FR_T = \left(\sum AERevents\right)/T_{sample} \tag{1}$$

$$FR_{channel\_i} = \left(\sum AERevents(i)\right)/FR_T \tag{2}$$

Figure 5 shows the normalized spike firing activity for a set of eight pure tones with frequencies that range from 130.813 Hz to 1396.91 Hz, logarithmically spaced.

The weights of the second step of the training phase are obtained from the firing output of each neuron in the hidden layer when using the set of audio samples as input after loading the weights calculated in the previous step into the connections between the input and the hidden layer. These firing outputs are normalized by dividing each of them by the maximum value. The results obtained are the weight values that will be used in the connections between the hidden and the output layer.

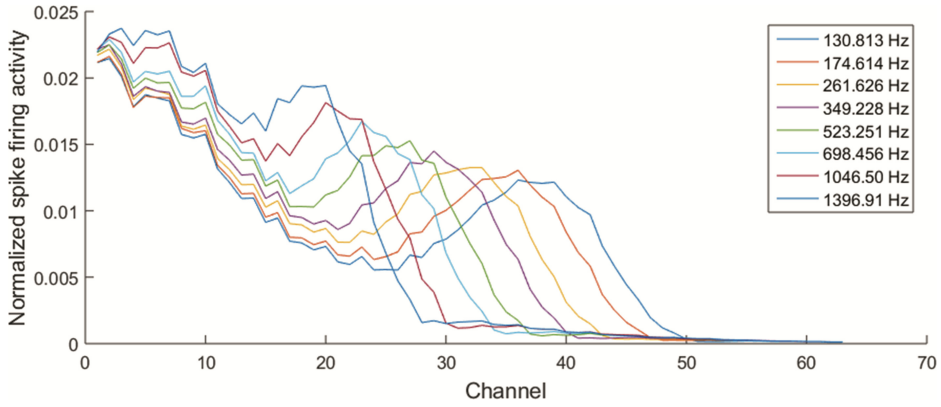


Fig. 5. Normalized spike firing activity for each NAS channel per audio sample.

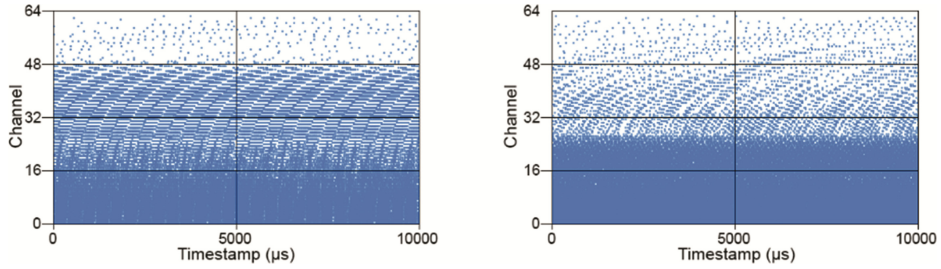


Fig. 6. First 10 ms cochleogram of the 130.813 Hz (left) and 1396.91 Hz (right) pure tones.

5 Test Scenario

In this work, the SNN architecture and training algorithm presented are tested using eight different audio samples. These output classes correspond to eight different pure tones with frequencies that range from 130.813 Hz to 1396.91 Hz, logarithmically spaced (130.813, 174.614, 261.626, 349.228, 523.251, 698.456, 1046.50 and 1396.91 Hz). These samples have a duration of 0.5 s and were generated using the audioplayer function from Matlab with a sampling rate of 48 KHz and a peak-to-peak voltage value of 1 V. After the signal is sent to the mixer, it propagates the sound to NAS input and sends an AER stream to the PC through the AER-USB interface. The jAER software running on the PC is able to capture this stream and save it as an aedat file. Figure 6 shows the cochleograms for the 130.813 Hz and the 1396.91 Hz pure tones after capturing them.

The first step of the training phase can be achieved by applying the equations presented in Sect. 4 to the set of eight aedat files corresponding to each pure tone. This will generate a CSV file containing the weights for the 64×8 connections between the input and the hidden layers of the SNN based on the firing rate of the spike streams for each audio sample. As described in the previous section, loading those weights into the corresponding connections and using the eight pure tones as input will result on a firing

output on the second layer neurons that will be used for training the connections between the second and the output layers of the SNN.

After the weights are set on these connections, new sets of the same pure tones (same frequencies) are recorded using different Signal-to-Noise Ratio (SNR) values and tested on the network, calculating the hit rate percentage for each class.

6 Experimental Results

Different pure tone sets with the same frequencies and properties (0.5 s and 0.5 V amplitude) as the ones used in this work were captured and used to test the network robustness and effectiveness. A 100 % hit rate was obtained for every class when the signal was a pure sine wave. Moreover, the network has also been tested by adding a noise signal consisting of random values to the pure tones original signals, obtaining audio samples with different SNR values (from 35.2 dB to 0 dB). The hit rate percentage for every class using the previous SNR values are listed in Table 1.

Table 1. Hit rate percentage of the audio samples classification SNN for different SNR values.

SNR (dB)	Pure tone frequency (Hz)							
	130.813	174.614	261.626	349.228	523.251	698.456	1046.5	1396.91
No noise	100 %	100 %	100 %	100 %	100 %	100 %	100 %	100 %
35.1993	100 %	100 %	100 %	100 %	100 %	100 %	100 %	100 %
21.3363	100 %	83 %	96 %	100 %	100 %	100 %	100 %	100 %
13.2273	100 %	81 %	92 %	100 %	100 %	100 %	100 %	96 %
7.4733	100 %	86 %	100 %	100 %	100 %	100 %	100 %	95 %
3.0103	74 %	90 %	100 %	98 %	100 %	100 %	100 %	98 %
2	93 %	88 %	20 %	32 %	16 %	92 %	32 %	97 %
1	10 %	5 %	0 %	0 %	0 %	88 %	26 %	94 %
0	0 %	0 %	0 %	0 %	0 %	76 %	22 %	91 %

The results show very high hit rate percentages when the SNR is above 3 dB. However, when the SNR falls below 3 dB and approaches zero dB (the amplitude of the pure tone is the same as the amplitude of the noise signal) the network is not able to classify every input signal as its corresponding class.

7 Conclusions

In this paper, a novel multilayer spiking neural network architecture for audio samples classification implemented in SpiNNaker has been presented. To achieve this goal, an optimized training phase for audio recognition has been described and specified in two different steps, which allow obtaining the weights for the connections between the input and the hidden layers and between the hidden and the output layers. The network was trained using eight pure tones with frequencies between 130.813 Hz and 1396.91 Hz and tested by adding a noise signal with SNR values between 35.1993 and 0 dB.

The hit rate values obtained after many tests confirm the robustness of the network and the training, which make it possible to classify every pure tone with a probability over 74 % even when the SNR value is 3 dB, obtaining almost a 100 % probability for every input when the SNR is above that value.

Finally, the SpiNNaker board has allowed to model and develop a leaky integrate-and-fire spiking neural network for this purpose in an easy, fast, user-friendly and efficient way, proving its potential, and promoting and facilitating the implementation of SNNs like these in real hardware platforms. The PyNN code used to test the SNN presented in this work is available at [19].

Acknowledgements. The authors would like to thank the APT Research Group of the University of Manchester for instructing us in the SpiNNaker. This work is supported by the Spanish government grant BIOSENSE (TEC2012-37868-C04-02) and by the excellence project from Andalusian Council MINERVA (P12-TIC-1300), both with support from the European Regional Development Fund.

References

1. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128 × 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. IEEE J. Solid-State Circ. **43**, 566–576 (2008)
2. Chan, V., Liu, S.C., van Schaik, A.: AER EAR: a matched silicon cochlea pair with address event representation interface. IEEE Trans Circ. Syst. I **54**(1), 48–59 (2007)
3. Häfliger, P.: Adaptive WTA with an analog VLSI neuromorphic learning chip. IEEE Trans. Neural Netw. **18**, 551–572 (2007)
4. Indiveri, G., Chicca, E., Douglas, R.: A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. IEEE Trans. Neural Netw. **17**, 211–221 (2006)
5. Jiménez-Fernández, A., Jiménez-Moreno, G., Linares-Barranco, A., et al.: Building blocks for spikes signal processing. In: International Joint Conference on Neural Networks, IJCNN (2010)
6. Linares-Barranco, A., et al.: A USB3.0 FPGA event-based filtering and tracking framework for dynamic vision sensors. In: Proceedings of IEEE International Symposium on Circuits and Systems, pp. 2417–2420 (2015)
7. Linares-Barranco, A., Gomez-Rodriguez, F., Jimenez-Fernandez, A., et al.: Using FPGA for visuo-motor control with a silicon retina and a humanoid robot. In: IEEE International Symposium on Circuits and Systems, pp. 1192–1195 (2007)
8. Jimenez-Fernandez, A., Jimenez-Moreno, G., Linares-Barranco, A., et al.: A neuro-inspired spike-based PID motor controller for multi-motor robots with low cost FPGAs. Sensors **12**, 3831–3856 (2012)
9. Hamilton, T.J., Jin, C., van Schaik, A., Tapson, J.: An active 2-D silicon cochlea. IEEE Trans. Biomed. Circ. Syst. **2**, 30–43 (2008)
10. Jimenez-Fernandez, A., Cerezuela-Escudero, E., Miro-Amarante, L., et al.: A binaural neuromorphic auditory sensor for FPGA: a spike signal processing approach. IEEE Trans. Neural Networks Learn. Syst. **1**(0) (2016)
11. Boahen, K.: Point-to-point connectivity between neuromorphic chips using address events. IEEE Trans. Circ. Syst II Analog Digit Sig. Process. **47**, 416–434 (2000)

12. Bekolay, T., et al.: Nengo: a Python tool for building large-scale functional brain models. Front Neuroinform. **7**, 48 (2014)
13. Goodman, D., Brette, R.: Brian: a simulator for spiking neural networks in python. Front Neuroinform. **2**, 5 (2008)
14. jAER Open Source Project. http://jaer.wiki.sourceforge.net
15. Berner, R., Delbruck, T., Civit-Balcells, A., Linares-Barranco, A.: A 5 Meps $100 USB2.0 address-event monitor-sequencer interface. IEEE International Symposium on Circuits and Systems (2007)
16. Painkras, E., et al.: SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation. IEEE J. Solid-State Circ. **48**, 1943–1953 (2013)
17. Davison, A.P.: PyNN: a common interface for neuronal network simulators. Front Neuroinform. **2**, 11 (2008)
18. SpiNNaker Home Page. http://apt.cs.manchester.ac.uk/projects/SpiNNaker
19. Dominguez-Morales, J.P.: Multilayer spiking neural network for audio samples classification using Spinnaker Github page. https://github.com/jpdominguez/Multilayer-SNN-for-audio-samples-classification-using-SpiNNaker

Appendix C

Deep Neural Networks for the Recognition and Classification of Heart Murmurs Using Neuromorphic Auditory Sensors

Authors

- Juan Pedro Dominguez-Morales
- Angel Jimenez-Fernandez
- Manuel J. Dominguez-Morales
- Gabriel Jimenez-Moreno

Publication

Title: IEEE Transactions on Biomedical Circuits and Systems

Type: Journal Article

Volume 12

Issue: 1

Publisher: IEEE

Date: February 2018

Pages: 24 - 34

ISSN: 1932-4545

Deep Neural Networks for the Recognition and Classification of Heart Murmurs Using Neuromorphic Auditory Sensors

Juan P. Dominguez-Morales, *Member, IEEE*, Angel F. Jimenez-Fernandez, *Member, IEEE*, Manuel J. Dominguez-Morales, and Gabriel Jimenez-Moreno, *Member, IEEE*

***Abstract*—Auscultation is one of the most used techniques for detecting cardiovascular diseases, which is one of the main causes of death in the world. Heart murmurs are the most common abnormal finding when a patient visits the physician for auscultation. These heart sounds can either be innocent, which are harmless, or abnormal, which may be a sign of a more serious heart condition. However, the accuracy rate of primary care physicians and expert cardiologists when auscultating is not good enough to avoid most of both type-I (healthy patients are sent for echocardiogram) and type-II (pathological patients are sent home without medication or treatment) errors made. In this paper, the authors present a novel convolutional neural network based tool for classifying between healthy people and pathological patients using a neuromorphic auditory sensor for FPGA that is able to decompose the audio into frequency bands in real time. For this purpose, different networks have been trained with the heart murmur information contained in heart sound recordings obtained from nine different heart sound databases sourced from multiple research groups. These samples are segmented and preprocessed using the neuromorphic auditory sensor to decompose their audio information into frequency bands and, after that, sonogram images with the same size are generated. These images have been used to train and test different convolutional neural network architectures. The best results have been obtained with a modified version of the AlexNet model, achieving 97% accuracy (specificity: 95.12%, sensitivity: 93.20%, PhysioNet/CinC Challenge 2016 score: 0.9416). This tool could aid cardiologists and primary care physicians in the auscultation process, improving the decision making task and reducing type-I and type-II errors.**

***Index Terms*—Audio processing, Caffe, convolutional neural networks, deep learning, heart murmur, neuromorphic sensor, pattern recognition.**

I. Introduction

HEART disease is a major health problem and is one of the main causes of death in the world. Cardiovascular disease (CVD) causes nearly half of the deaths in Europe (48%) [1] and 34.3% in America (1 in 2.9 deaths in the United States) [2]. Detecting CVDs at an early stage is crucial for applying the corresponding treatment and reduce the potential risk factors. Auscultation is one of the most used techniques for this purpose, and can provide clues to the diagnosis of many cardiac abnormalities by listening and analyzing the heart sound components using a stethoscope. It is very cheap and requires minimal equipment. However, physicians need extensive training and experience for auscultating [3]. Moreover, the accuracy rate of primary care physicians and medical students on the auscultation process is between 20–40%, as reported in [4]–[7], and only roughly 80% is achieved by expert cardiologists [4], [7], [8].

Heart murmurs are sounds produced when blood flows across one of the heart valves that are loud enough to produce audible noise. Murmurs may be harmless (innocent), which are primarily due to physiologic conditions outside the heart, or abnormal, which may be a sign of a more serious heart condition or a structural defect in the heart itself. The most common problems that cause abnormal heart murmurs are mitral or aortic stenosis and mitral or aortic regurgitation. The sounds can also be categorized by timing, into systolic and diastolic, differing in the part of the heartbeat on which they can be heard (between the S1 and S2 heart sounds, or starting at or after S2 and ending before or at S1, respectively).

Heart murmurs are the most common abnormal finding when a patient visits the physician for auscultation. A heart murmur does not necessarily lead to having a CVD; it could be an innocent murmur instead of a pathological one, which does not represent current or future illness. The physician must decide if the patient is healthy or not, but, due to the fact that the accuracy is not great, the expert could be wrong, making type-I or type-II errors. A type-I error (alpha error) is the detection of an effect that is not present (i.e., healthy patients are sent for echocardiogram), while a type-II error (beta error) is failing on the detection of an effect that is present (i.e., pathological patients are sent home without medication or treatment). It is clear that, in this case, type-II errors are more important to avoid.

Manuscript received April 8, 2017; revised August 3, 2017; accepted September 5, 2017. Date of publication September 22, 2017; date of current version January 26, 2018. This work was supported by the Spanish government under Grant (with support from the European Regional Development Fund) COFNET (TEC2016-77785-P). The work of J. P. Dominguez-Morales was supported by a Formación de Personal Universitario Scholarship from the Spanish Ministry of Education, Culture and Sport. This paper was recommended by Associate Editor S. Renaud. *(Corresponding author: Juan P. Dominguez-Morales.)*

The authors are with the Robotic and Technology of Computers Laboratory, Department of Architecture and Technology of Computers, University of Seville, Seville 41012, Spain (e-mail: jpdominguez@atc.us.es; ajimenez@atc.us.es; mdominguez@atc.us.es; gaji@atc.us.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBCAS.2017.2751545

TABLE I
COMPARATIVE STUDY BETWEEN STATE-OF-THE-ART STUDIES ABOUT HEART SOUND DIAGNOSIS SYSTEMS

Ref.	Preprocessing	Classification method	Classes	Results	No. of samples
[9]	-Segmentation -Alignment -Spectrogram	ANN[1]	3: normal, AS[a] or AR[b]	85% (using simulated heart sounds) 48.7% (using real ones)	Train: 24 per class Test: 7 normal, 4 AS[a], 2 AR[b]
[10]	-Lowpass filtering -Segmentation (manually) -S-Transform	MLP[2] ANN[1]	5: normal, AS[a], AR[b], MS[c] or MR[d]	98% (using simulated heart sounds)	Train: 30 per class Test: 20 per class
[12]	-Wavelet transform -Normalization -Segmentation -Normalized Average Shannon Energy -Envelope extraction algorithm	FNNSL[3]	2: normal or abnormal	100% (using real heart sounds)	Train: 1 normal and 10 abnormal Test: 2 normal and 2 abnormal
[14]	-Segmentation -Trimmed Mean Spectrogram	PNN[4]	2: normal or abnormal	96.3% (using real heart sounds)	Train: 25 non-pathological, 36 pathological Test: 18 non-pathological, 37 pathological
[15]	-Band-pass filtering -Short-time Fourier transform -Features manually extracted from spectrogram -Interesting areas were manually selected	Statistical analysis	2: innocent or pathological	90.9% (using real heart sounds)	447 innocent, 272 pathological
[16]	-Short-time Fourier transform -Mean value of the signal segments -Band-pass filtering -Segmentation	SVM[5]	2: innocent or pathological	96.07% (using real heart sounds)	Train: 20 innocent, 20 pathological Test: 5 innocent, 5 pathological
[17]	-Wavelet decomposition -Feature detection using MIR toolbox from Matlab	Statistical analysis	7: normal, PDA[e], PS[f], MR[d], ASD[g], MS[c] or TOF[h]	85.08% (using real heart sounds)	20 normal, 9 PDA[e], 6 PS[f], 12 MR[d], 13 ASD[g], 17 MS[c], 13 TOF[h]
[18]	-Resampling -Band-pass filtering -Segmentation	AdaBoost + CNN[6]	2: normal or abnormal	94.24% sensitivity and 77.81% specificity. PhysioNet/CinC Challenge 2016 score: 86.02% (using real heart sounds)	Train: 2575 normal, 665 abnormal Test: 984 normal, 153 abnormal (PhysioNet/CinC Challenge 2016 dataset)
[19]	-18 features extracted from time, frequency and time-frequency domains based on a wrapper feature selection scheme.	Ensemble of SVMs[5]	2: normal or abnormal	86.91% sensitivity and 84.90% specificity. PhysioNet/CinC Challenge 2016 score: 85.90% (using real heart sounds)	Train: 2301 normal, 570 abnormal Test: 984 normal, 153 abnormal (PhysioNet/CinC Challenge 2016 dataset)

[1]: Artificial Neural Network. [2]: Multilayer Perceptron. [3]: Fuzzy Neural Network with Structure Learning. [4]: Probabilistic Neural Network. [5]: Support Vector Machine. [6]: Convolutional Neural Network.
[a]: Aortic Stenosis. [b]: Aortic Regurgitation. [c]: Mitral Stenosis. [d]: Mitral Regurgitation. [e]: Patent Ductus Arteriosus. [f]: Pulmonary Stenosis.
[g]: Atrial Septal Defect. [h]: Tetralogy of Fallot.

However, echocardiograms cost between $750 and $1500 [4] per patient, making type-I errors also important to avoid. The probability of needing this costly procedure could be reduced for both healthy people and pathological patients if a reliable (with a high accuracy rate) diagnostic tool were available as an aide for physicians.

The classification of heart sounds is not a new topic. Many studies have worked toward designing practical murmur classifier systems to improve the diagnostic accuracy of physicians. Most of them use neural networks (NNs), support vector machines (SVMs) or some complex preprocessing algorithms to carry out this task [7]–[17], [18], [19]. Many studies like [10], [11], [15] have used a processing step where a person selects the best portion of the sound signal that should be used as input to the system, making this solution not ideal for a real scenario because of the need of human interaction. Some of them have used NNs to classify between different kinds of heart murmurs [7], [9]–[11], but have only trained the network with simulated heart sounds with no noise, obtaining very bad accuracy results when testing the classifier with real heart sounds (48.5%). Others have used only a small amount of real heart sounds [10], [12], [14]–[17], which is not representative when it comes to testing it in a real scenario. Table I summarizes the main information about the preprocessing and the classification steps that have been performed in some of the state-of-the-art studies that have been discussed in this section, along with the two leading approaches from the PhysioNet/CinC Challenge 2016. Works like [20] use similar preprocessing techniques and classification algorithms, but focusing on cough sounds identification instead of heart murmurs.

The main aim of this work is to develop a classifier system using a Convolutional Neural Network (CNN) that accepts heart sound recordings directly after preprocessing the information, and classifies the input to identify if the person whose heart sound is acquired, is either a healthy person or a pathological patient. The preprocessing step automatically divides the heart sound recordings into windows of a specific time length. Heart murmurs are located in the 195 Hz band [7], [9], but can reach up to 700 Hz [10], [21], which confirms that they can be identified and extracted from the heart sound signal in the frequency domain. For this purpose, these segments of the original sound are sent to a Neuromorphic Auditory Sensor (NAS) [22], which tries to mimic the way in which the inner ear works, decomposing the audio into frequency bands, and packetizes the information using the Address-Event Representation (AER) communication protocol [23]. Then, this information is converted to sonogram images, which are then used as input to the CNN for further classification using deep learning algorithms.

The rest of the paper is structured as follows: Section II presents an overview of the system architecture using a block diagram to explain each of the components in it. Then, Section III describes the Neuromorphic Auditory Sensor (NAS) [22] and how its output information is saved into AEDAT files [24] in the computer using a USBAERmini2 board [25]. After this, in Section IV, the dataset acquisition is explained, describing the heart sound database that has been used in this work. Section V presents the preprocessing algorithms executed using the data before applying them as input to the classifier system. **Caffe** [26], which is one of the most used deep learning frameworks, is described in Section VI along with the Convolutional Neural Networks (CNNs) that have been trained and tested in order to classify the heart sound dataset. Section VII presents the classification results and the comparison between the different experiments that have been carried out in this work. Finally, the conclusions of this work are presented in Section VIII.

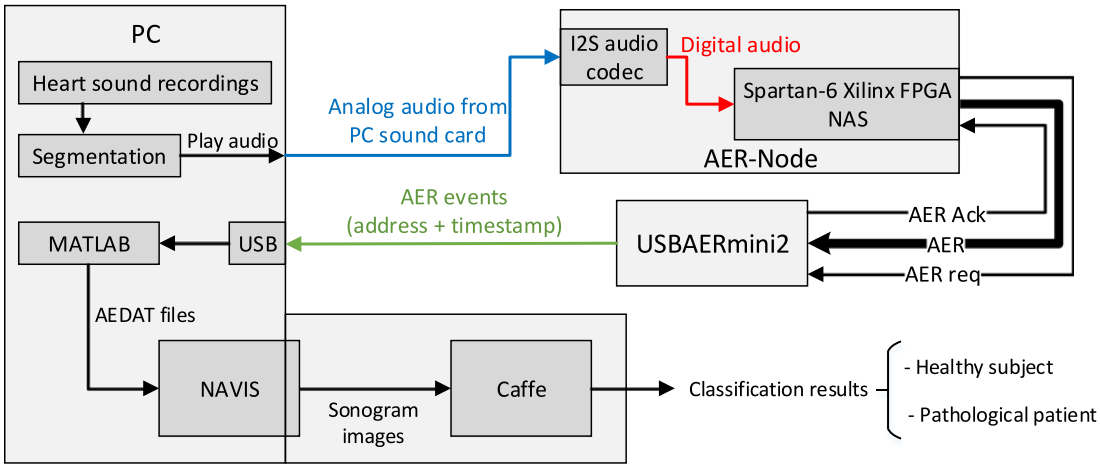


Fig. 1. Block diagram of the system architecture.

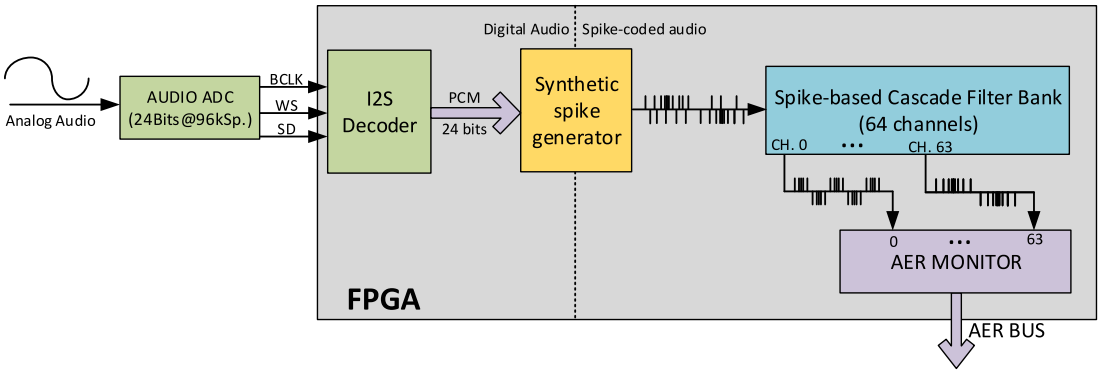


Fig. 2. Mono-aural Neuromorphic Auditory Sensor for FPGA with an I2S audio ADC and AER interface.

II. System Overview

The system consists of different modules and steps to achieve its purpose. Most of them are carried out in the computer; however, one of the most important parts of the preprocessing is done outside of it, on a Field Programmable Gate Array (FPGA). A block diagram of the whole system is presented in Fig. 1.

The heart sound recordings used in this work are obtained from the PhysioNet/CinC Challenge database [27]. It consists of 3,126 heart sound recordings, lasting from 5 to over 120 seconds. The main idea is that, after some preprocessing functions are applied to the information, one image is obtained for each of the audio samples contained in the dataset, so that it could be used as input to feed the CNN. In order to generate images with the same width and height, the first preprocessing step is to divide the heart sound recordings into segments of the same length. In this work, the accuracy of the system has been tested using segmentation windows of 1, 1.25 and 1.5 seconds (without overlapping), which were chosen because they are large enough to contain the information from, at least, one full cardiac cycle, but at the same time small enough to generate as many samples as possible.

After this process is completed (generating 77573, 61518 and 51009 samples when using a segmentation window of 1, 1.25 and 1.5 seconds length, respectively), audio samples are sent to the audio input of an AER-Node platform [28]. A 64-channel mono NAS (Neuromorphic Auditory Sensor) [22] is programmed on the Spartan-6 FPGA that the AER-Node board has, which decomposes the audio signal into frequency bands and packetizes the information using the AER (Address-Event Representation) protocol [23]. An USBAERmini2 board [25] receives this information and sends it to the computer through a USB port. Then, a script running on MATLAB collects the AER packets received and stores them into AEDAT files [24] (one file per audio sample), which is the standard format used for storing this kind of information.

A grayscale sonogram image is generated for each AEDAT file using Neuromorphic Auditory VISualizer Tool (NAVIS) [29], which is a desktop software application that is able to load AEDAT files and postprocess the information obtained from the NAS, generating useful charts like the cochleogram, sonogram, histogram, etc. The whole set of images obtained are then divided into three different datasets: one for training the

CNN (75% of the total amount of images), a second one for validation (15%) and the last one to test the CNN and obtain the accuracy ratio of the system (10%). Different CNN models have been trained and tested using **Caffe** and their accuracy results have been compared. Each of these elements and steps will be described in detail in the next sections.

III. Neuromorphic Auditory Sensor

Neuromorphic Auditory Sensor (NAS) is an audio sensor for FPGAs inspired by Lyon's model of the biological cochlea [30]. This sensor is able to process an excitatory audio signal using Spike Signal Proccesing (SSP) techniques [31], decomposing incoming audio in its frequency components, and providing this information as a stream of events using the Address-Event Representation (AER) [23]. Current state-of-the-art of silicon cochleae process audio in an analog way [32], using a bank of low-pass filters (modeling the basilar membrane), and convert the filters' output to spikes (modeling the inner hair cells). However, NAS works in the opposite way: first, it converts the incoming audio to spikes, and directly processes these spikes using a Spike Low-pass Filter (SLPF) bank with a cascade topology. Due to the use of SSP filters, circuits are very simple and do not need complex operating units or dedicated resources (e.g. floating point ALUs, hardware multipliers, RAM memory, etc...). As a consequence, NAS designers are able to replicate SLPFs in low-cost FPGAs, building large scale NAS with a low-clock frequency working fully in parallel.

To digitalize audio signals we use a commercial analog-to-digital audio converter (CS5344, with a resolution of 24 bits and a sample rate of 96 kSamples/sec.), that provides the audio samples using an I2S bus. Inside the FPGA, audio samples from the I2S bus are decoded to 24 bits digital words with two's complement. Digital audio samples are written in a synthetic spike generator (SSG), which provides a spike stream with a frequency that is proportional to the digital amplitude. These spikes are used as input to a bank of 64 SSP filters with a cascade topology, known as Cascade Filter Bank (CFB), which processes audio spikes decomposing them in frequency. Finally, output spikes from CFB are connected to an AER-Monitor [33]. This gives a unique address to the fired spikes following the Address-Event Representation, and propagates them using an asynchronous AER bus. Fig. 2 shows the block diagram of the architecture of a mono-aural NAS.

A 64-channel mono-aural NAS for FPGA with a cascade topology has been used together with a USB-AERmini2 interface [25], as can be seen in Fig. 3. NAS response is stored as AEDAT files and the output information can be seen in the second image (b) of Fig. 4, where each dot corresponds to an event that has been fired in a particular AER address at a specific time.

IV. Dataset Acquisition

The heart sound dataset used in this work contains the recordings used in the PhysioNet/CinC Challenge 2016 [34], [27], which comprises nine heart sound databases from different research groups. Heart sound recordings were sourced from several contributors around the world from both healthy subjects

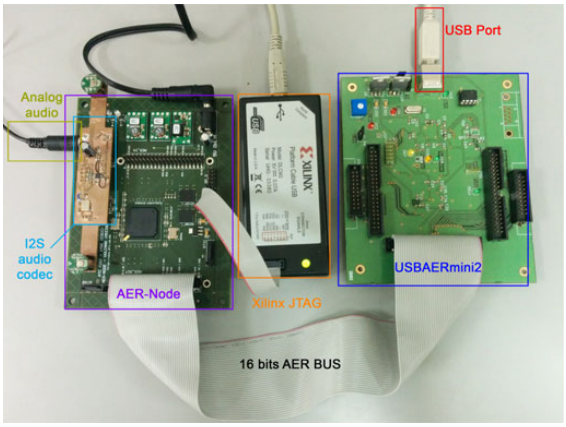


Fig. 3. NAS connected to an USBAERmini2.

and pathological patients including children and adults, and contains a total of 3,126 heart sound recordings, lasting from 5 seconds to over 120 seconds. The heart sound recordings were collected from different locations on the body: aortic area, pulmonic area, tricuspid area and mitral area. These recordings are divided into two types: normal and abnormal heart sound recordings. The normal recordings were from healthy subjects and the abnormal ones were from patients with a confirmed cardiac diagnosis, which is not specified, but typically they are coronary artery diseases and heart valve defects like mitral valve prolapse, mitral regurgitation, aortic stenosis and valvular surgery.

Audio recordings were resampled to 2000 Hz and have been divided into three different sets of mutually exclusive populations, using 75% of them to train the network, 15% for validation and 10% to test the network. These recordings are not clean and contain noise from various sources due to the uncontrolled environment, such as talking, breathing, stethoscope motion and intestinal sounds, which is important to note because training the system with these real sounds will make it more robust and noise tolerant.

Using only 75% of the samples that this dataset has (which is only a total of 2345 heart recordings) for training the CNN is not sufficient if we want our system to be robust enough for a test with different recordings that are not included in that collection. Moreover, working with audio files with variable lengths is neither appropriate nor optimal for training a CNN: dividing these files into shorter ones (in terms of duration) would generate more samples that could be used to both train and test the network, making the system more reliable. For this purpose, the heart recordings obtained from the PhysioNet dataset were segmented using a fixed window length. The segmentation is one of the steps that have been carried out in the preprocessing phase, which is described in the next section.

V. Preprocessing of the Information

Sound recordings from the PhysioNet database do not have the same length (each file lasts from 5 to 120 seconds) and CNNs need the input images to have the same width and height for

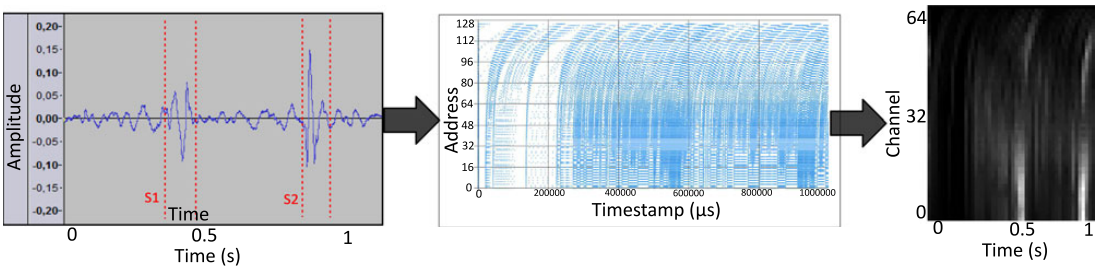


Fig. 4. Outputs of the different preprocessing steps: the first image (a) is the original audio signal after the segmentation process; the second one (b) is the AER information obtained from the NAS' output; and the last one (c) is the grayscale sonogram image obtained with NAVIS, where a whiter tone in a specific section means that that section has more activity. (a) Audio signal. (b)Cochleogram. (c) Sonogram.

Algorithm 1: Sonogram calculation.

```
integPeriod = 20 ms
sonogram = zeros(max(in_addr), max(in_tStamp)/
integPeriod)
for i = 1: max(in_addr) do
    sonogram(in_addr(i), in_tStamp(i)/integPeriod)++
end for
```

training and testing the network. For this purpose, a segmentation algorithm is applied to each of the samples before sending the audio signal to the NAS' audio input connector. In this work, different experiments have been carried out, using 1, 1.25 and 1.5 second-long windows in the segmentation process, obtaining 77573, 61518 and 51009 samples, respectively. This way, the number of samples available is also increased (more than 16 times the amount of samples in the default heart recordings database), which will provide more information in the training process of the CNN (these algorithms need a huge amount of images to train the system more robustly). Each of these three datasets has been used to feed different CNN models and the classification results are presented in Section VII.

These length values were selected due to the fact that they can contain the information from a full cardiac cycle at least (from the phase of relaxation diastole to the phase of contraction systole; or, in terms of sound, the whole "lub-dub" sequence including S1 and S2).

As was presented in the introduction (Section I), heart murmurs are located in the 195 Hz band [9], but can reach up to 700 Hz [21], which confirms that they can be identified and extracted from the heart sound signal in the frequency domain. For this purpose, each of the audio segments obtained from the original sound in the previous step are sent to a NAS, which mimics the way in which the inner ear works, decomposing the audio into frequency bands, and packetizes the information using the AER communication protocol. These packets are sent to the computer through a USB port using the USBAERmini2 board. A script in MATLAB is then used to generate an AEDAT file, which is the standard format used for storing this kind of information, for each of the audio samples. These files contain information about the address and timestamp of every event that has been fired in the NAS when feeding its input with an analog audio signal.

NAVIS is a GPL-licensed desktop software application that allows to post-process the information obtained from a NAS. This tool implements a set of charts that allows to represent the auditory information as cochleograms, histograms and sonograms, among others. It can also split the auditory information into different sets depending on the activity level of the spike streams. Due to the open-source nature of the project [35], it has been modified to automatically take the AER information contained in the AEDAT files that were obtained after sending each of the segmented samples to the NAS, and generate grayscale sonogram images based on the activity levels of the sound recordings in the frequency domain across the NAS' channels.

The pseudocode shown in Algorithm 1 presents the algorithm that has been used to calculate the sonogram's matrix of values (pixels of the image). These values are then normalized between 0 and 255, and a grayscale tone is set based on each value (0 being black, and 255 being white). Image (c) in Fig. 4 shows the output sonogram from one of the 1 second-long heart sound recordings.

The whole preprocessing step can be seen in Fig. 4. The first image (a) shows the audio signal that corresponds to one of the 1-second samples after being segmented from the original heart recording. Then, the second one (b) is the cochleogram of the information contained in the AEDAT file that was obtained after sending the audio signal to the NAS and capturing the output information using MATLAB and the USBAERmini2 board. Each dot of the cochleogram is an event that has been fired for a particular AER address (there are 128 addresses in a 64-channel mono NAS: each channel has two addresses, for positive and negative spikes) at a specific time (timestamp). The sonogram of the AEDAT file (c) was calculated using the equation that was previously described, resulting in a grayscale image with a width of 50 pixels (using time windows of 20000 μs in length for integrating the information) and a height of 64 pixels (the number of both negative and positive spikes from the same channel add up).

VI. Caffe

Caffe (Convolutional Architecture for Fast Feature Embedding) is a customizable framework for state-of-the-art deep learning algorithms. It allows to train and deploy general pur-

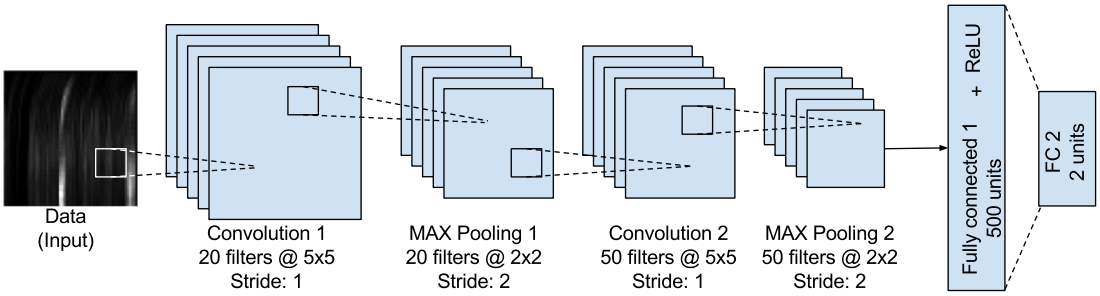


Fig. 5. Block diagram of the LeNet-5 model architecture.

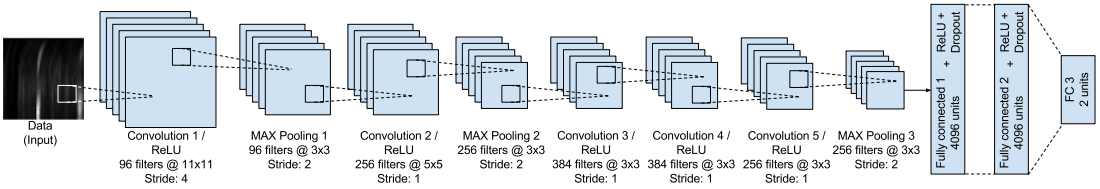


Fig. 6. Block diagram of the AlexNet model architecture.

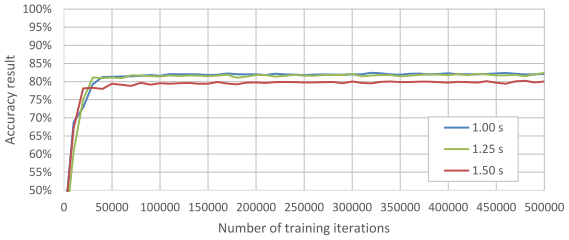


Fig. 7. Accuracy results achieved for each dataset (1s in blue, 1.25 s in green and 1.5 s in red) per 10000 training iterations using the default LeNet-5 model. Accuracy ratios obtained after 500000 training iterations: 82.11%, 82.39% and 80.00%, respectively.

pose CNNs and other deep models efficiently and in an easy way. **Caffe** is capable of processing over 40 million images a day on a single K40 or Titan GPU (∼2.5 ms per image) thanks to CUDA GPU computation. It has been used in many research fields like vision, speech recognition, robotics, neuroscience and astronomy.

Caffe provides a complete toolkit for training, testing and deploying models, which can be described using the BSD-licensed C++ library with Python and Matlab bindings. The framework also provides a collection of reference models and well-documented examples for all of these tasks, including the "AlexNet" ImageNet model [36] and the "LeNet" MNIST model [37]. These models can be modified, allowing to add/remove layers to/from the network, change the input dataset format and train it with different activation functions and parameters, which are already implemented. **Caffe** model definitions are written using the Protocol Buffer language [38], which is a language-neutral platform-neutral and easy to use mechanism for serializing structured data.

In this work, a modified version of the LeNet-5 CNN [37] has been used, where the number of outputs has been changed to two, as the goal is to distinguish between two classes: healthy subject and pathological patient. This model was designed for handwritten and machine-printed character recognition, but it is also well known for its high accuracy results for image recognition and feature extraction. Many studies have used this model for a wide variety of purposes, like freehand sketch recognition [39], Alzheimer's disease recognition [40] or even horse gait classification [41], obtaining very good results.

Fig. 5 shows the block diagram representation of the LeNet-5 model. The input dataset and the input image size have been set to match our requirements.

Several tests have been performed, using different values on some of the parameters of the Solver Prototxt file (which is the file that contains the network's training configuration) for each of the three datasets that were obtained after the preprocessing step (using 1 second, 1.25 seconds and 1.5 seconds audio length windows on the segmentation phase). The parameters that have been changed from the Solver Prototxt file are: (1) the base learning rate of the network (base_lr); (2) the momentum, which indicates how much of the previous weight will be retained in the new calculation (momentum); (3) the weight decay, which is the factor of penalization of large weights (weight_decay); (4) the test interval, which has been set to 10000 training iterations (test_interval); (5) the number of test iterations that should occur per test_interval (test_iter), to match the number of samples that the dataset has; and (6) the maximum training iterations, indicating when the network should stop training, which has been set to 500000 (max_iter). These parameters were optimized by repetition and comparison. The solver mode has been changed from CPU to GPU, due to the fact that the training process has been carried out using a NVIDIA GeForce GTX 1060 with 6GB of GDDR5 memory, and CUDA Toolkit 8.

TABLE II
TRAINING PARAMETERS AND LAYER CONFIGURATIONS FOR EACH OF THE CNNS USED

	base learning rate	learning policy	momentum	weight decay	Conv. layers	Pool. layers
Default LeNet-5	0.01	inv	0.9	0.0005	-Kernel sizes: 5 and 5 -Strides: 1 and 1	-Kernel sizes: 2 and 2 -Strides: 2 and 2
Modified LeNet-5	0.013	inv	0.6	0.000875	-Kernel sizes: 3 and 3 -Strides: 1 and 1	-Kernel sizes: 2 and 2 -Strides: 1 and 1
Default AlexNet	0.01	step (10000 iter)	0.9	0.0005	-Kernel sizes: 11, 5, 3, 3 and 3 -Strides: 4, 1, 1, 1 and 1	-Kernel sizes: 3, 3 and 3 -Strides: 2, 2, and 2
Modified AlexNet	0.013	step (10000 iter)	0.6	0.000875	-Kernel sizes: 3, 3, 3, 3 and 3 -Strides: 2, 1, 1, 1 and 1	-Kernel sizes: 3, 3 and 3 -Strides: 1, 1, and 1

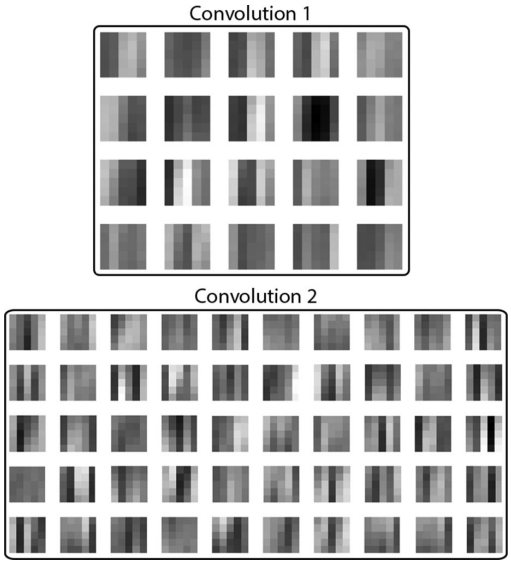


Fig. 8. Features learned for the two convolution layers (20 and 50 filters, respectively) with the default version of the LeNet-5 model.

Other CNNs like the AlexNet (Fig. 6), which is a much more complex network, has also been tested for this purpose and the accuracy results and comparison between this and the LeNet-5 models are presented in the next sections.

VII. RESULTS AND DISCUSSION

Three different window lengths have been used in the segmentation process in this work: 1, 1.25 and 1.5 seconds. As presented in the section where the preprocessing of the information is described, using these three sample lengths leads to obtaining up to 77573, 61518 and 51009 samples, respectively, which is enough for training and testing a CNN. In this work, modified versions of two widely-known CNN models have been used. The accuracy of the network has been obtained for each of the experiments. The sensitivity (Se), specificity (Sp) and the PhysioNet/Computing in Cardiology Challenge 2016 score (MAcc) have been calculated for the approaches that achieved the best accuracy results using the equations that are defined in [42].

A. Using the LeNet-5 Model

First, the accuracy of the system was tested using the LeNet-5 model [37]. The architecture of the model is presented in Fig. 5: it consists of a convolutional layer followed by a pooling layer, another convolutional layer followed by a pooling layer, and then two fully connected layers similar to the conventional multilayer perceptrons. The classifier was trained and tested using each of the three datasets described before without applying any modification to the training parameters or to the configuration of the CNN's layers. The accuracy results can be seen in Fig. 7 for every 10000 training iterations up to a total of 500000 using a base learning rate of 0.01, the *inv* learning policy, 0.9 as momentum and 0.0005 as weight decay. The *inv* learning policy updates the learning rate based on the equation shown in (1), where *gamma* is set to 0.0001 and *power* to 0.75. Table II summarizes the training parameters and layer configurations (kernel sizes and strides for each convolution and pooling layer) for each of the CNN models used in this work.

$$l\_rate = l\_rate * (1 + gamma * iter)^{(-power)} \quad (1)$$

After the default LeNet-5 CNN was trained and tested, 82.11% was achieved for the 1-second dataset, 82.39% for the 1.25-seconds dataset, and 80.00% for the 1.5-seconds dataset. Se, Sp and MAcc were calculated for the dataset that achieved the best accuracy, obtaining 83.26%, 78.58% and 0.8092, respectively. Even though the model was not modified from its default state to improve the classification, the obtained results were very similar to the accuracy that expert cardiologists are able to achieve when auscultating. Fig. 8 shows the features that the default LeNet-5 model is learning on each of its convolution layers. It can be seen that the first layer extract vertical information from the images and the second one is able to detect more complex patterns. However, the results obtained could be improved by changing the network configuration.

In this context, the next experiment consisted in modifying the same CNN model and its training parameters to improve the accuracy results of the system. As in the previous case, the input layer was adapted to be able to work with the proper image size that matches its corresponding dataset (50x64 for the 1 s sample length dataset, 63 $\times$ 64 for the 1.25 s dataset and 75 $\times$ 64 for the 1.5 s dataset). Moreover, kernel sizes were reduced from 5 to 3 and the stride from 2 to 1, for a more detailed analysis of the input images, which allows the extraction of more features

Fig. 9. Accuracy results achieved for each dataset (1s in blue, 1.25 s in green and 1.5 s in red) per 10000 training iterations using the modified version of the LeNet-5 model. Accuracy ratios obtained after 500000 training iterations: 93.68%, 93.57% and 91.14%, respectively, which are better that the ones obtained previously.

from them. Training parameters were optimized by repetition and comparison until the best results were obtained for each of the datasets.

Fig. 9 presents the accuracy results for every 10000 training iterations up to a total of 500000 using a base learning rate of 0.013, the *inv* learning policy, 0.6 as momentum and 0.000875 as weight decay. As can be seen, the 1 s dataset achieves the best result (93.68%), while the 1.25 s and the 1.5 s datasets achieve 93.57% and 91.14% accuracy ratios, respectively. The chart also shows that using smaller window length values in the segmentation step makes the network take a higher number of iterations to converge when training the CNN, due to the fact that more images are generated in the process. Se, Sp and MAcc were calculated for the 1.25 s dataset, obtaining 92.84%, 91.48% and 0.9216, respectively. Training the system took an average of four hours to complete when using the default model, and six hours ($\sim$375 minutes) for the modified model, for each of the experiments and datasets with a NVIDIA GeForce GTX 1060 GPU. The first approaches were carried out using the CPU (3.2 GHz Intel i5-4460) instead of the GPU, which increased the training process execution time more than 24 hours. An average of 13.7% improvement over the default LeNet-5 model was achieved in this case.

B. Using the AlexNet Model

The same experiments that were performed using the LeNet-5 model were then tested with a more complex architecture: the AlexNet [36]. The network is made up of 5 convolutional layers, max-pooling layers, dropout layers and 3 fully connected layers. It was released in 2012 by Alex Krizhevsky and scaled the insights of the LeNet-5 model into a much deeper and wider neural network that could be used to learn much more complex objects. It was used to win by a large margin the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) [43].

First, the network was trained and tested without modifying the architecture or the training parameters (only the input and output layers were adapted to accept the image sizes that are being used in this work, and to classify between two different categories). The accuracy results can be seen in Fig. 10, where a base learning rate of 0.01 is used along with the *step* learning policy and 0.9 and 0.0005 as momentum and weight decay,

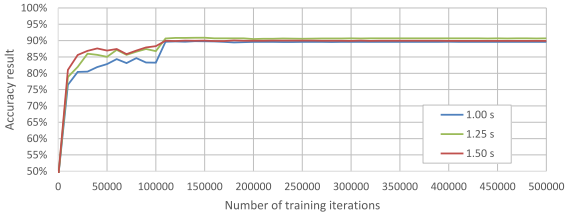


Fig. 10. Accuracy results achieved for each dataset (1s in blue, 1.25 s in green and 1.5 s in red) per 10000 training iterations using the default version of the AlexNet model. Accuracy ratios obtained after 500000 training iterations: 89.61%, 90.70% and 89.91%, respectively.

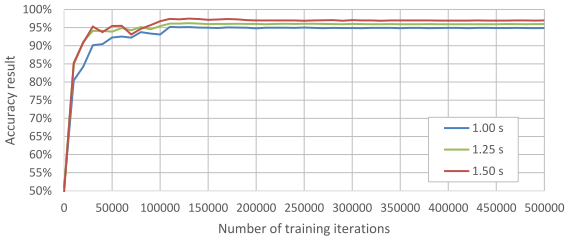


Fig. 11. Accuracy results achieved for each dataset (1s in blue, 1.25 s in green and 1.5 s in red) per 10000 training iterations using the modified version of the AlexNet model. Accuracy ratios obtained after 500000 training iterations: 94.88%, 95.95% and 97.05%, respectively.

respectively. Se, Sp and MAcc were calculated for the dataset that achieved the best accuracy, obtaining 94.52%, 90.48% and 0.9250, respectively. As can be seen, the results do not differ much from the ones obtained with the modified version of the LeNet-5 model while using the default training parameters. Other learning policies like *fixed* and *inv* (which is the one that the LeNet-5 model uses) were used without modifying the rest of the network, but the results did not improve significantly. The *step* learning policy updates the learning rate based on the equation shown in (2), where *gamma* is set to 0.1 and *step* to 100000.

$$l\_rate = l\_rate * gamma^{(floor(iter/step))} \tag{2}$$

In the next experiment, the AlexNet model was modified, reducing kernel sizes and the stride value for each convolutional layer. Training parameters were changed to the ones with whom the LeNet5 obtained the best results, and, after that, they were optimized by repetition and comparison. Fig. 11 presents the accuracy results for every 10000 training iterations up to a total of 500000 using a base learning rate of 0.013, the *step* learning policy, 0.6 as momentum and 0.000875 as weight decay. In this case, the 1.5 s dataset achieved the best result (97.05%), while the 1s and the 1.25 s datasets achieved 94.88% and 95.95% accuracy ratios, respectively. This could be due to the fact that training a more complex CNN like the AlexNet allows to extract more information from the 1.5 s images, which was not possible with the LeNet-5 model. Se, Sp and MAcc were calculated for the dataset that achieved the best accuracy, obtaining 95.12%, 93.20% and 0.9416, respectively.

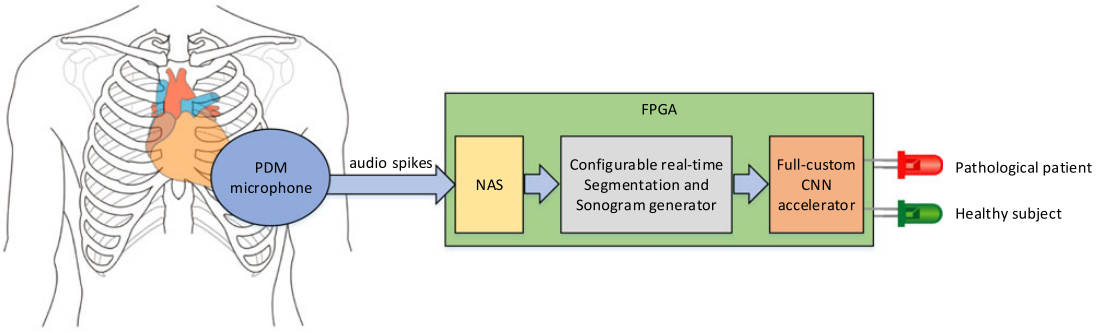


Fig. 12. Block diagram of the complete system implemented on an FPGA using a PDM microphone for real-time analysis of the heart sound directly from the patient.

TABLE III
ACCURACY, SENSITIVITY, SPECIFICITY AND PHYSIONET/CINC CHALLENGE 2016 SCORE OF THE DIFFERENT STUDIED APPROACHES

	Accuracy	Sensitivity(*Se*)	Specificity(*Sp*)	MAcc
Primary care physicians	40%	–	–	–
Expert cardiologists	80%	–	–	–
[18] Potes *et al.*	–	94.24%	77.81%	0.8602
[19] Zabihi *et al.*	–	86.91%	84.90%	0.8590
Default LeNet-5	82.39%	83.26%	78.58%	0.8092
Modified LeNet-5	93.68%	92.84%	91.48%	0.9216
Default AlexNet	90.70%	94.52%	90.48%	0.9250
Modified AlexNet	97.05%	95.12%	93.20%	0.9416

Best cases for the 1, 1.25 and 1.5 datasets are selected.

An average of 65 hours for the default model and 107 hours for the modified model were needed to train the AlexNet CNN using the GPU. The CPU was intended to be used instead of the GPU in the first place, but the training process was estimated around three months (for the default version) to complete per experiment, which is an unreasonable amount of time. However, as can be seen in the images, the system converges after the first 150000 training iterations, approximately, which corresponds to 20 and 32 hours, respectively. Hence, the whole system could be trained for less than half of the iterations and obtain a very similar accuracy while spending much less time in the training process.

The modified version of the AlexNet model achieved the best results. However, it is important to point out that this CNN only improves the accuracy of the modified version of the LeNet-5 (which is a much simpler CNN model) by around 3.5%, while taking almost eighteen times the time needed to train the second one.

VIII. CONCLUSION

In this work the authors have presented a useful tool to aid cardiologists and primary care physicians in the auscultation process. The system uses heart sound recordings from both healthy patients and pathological patients directly, which are first split using windows with a fixed length (1, 1.25 and 1.5 seconds) and then sent to a NAS where the frequency components of the audio are extracted. After this, sonogram images are generated for each of the samples using NAVIS. These images were used to feed different CNN models (LeNet-5 and AlexNet) capable of extracting interesting features from them, which have been trained and tested with different configurations in **Caffe** to classify between the two categories that were described.

The obtained results using different LeNet-5 and AlexNet configurations achieve up to 97.05% accuracy rate in the best case (with a modified version of the AlexNet model), and 80.00% in the worst case (with the default LeNet-5 configuration). These accuracy rates include the 80% accuracy level of an expert cardiologist (see Table III for a comparative study of the obtained results), proving that the system could be very useful as an aide for cardiologists and primary care physicians in the auscultation process, reducing the number of both type-I and type-II errors made. Thereby, the authors have presented a reliable diagnostic tool that could improve the detection of pathological heart murmurs when auscultating and, by aiding the physician, achieve almost 100% accuracy between both. Also, the results have been compared in terms of sensitivity, specificity and the PhysioNet/Computing in Cardiology Challenge 2016 score (obtaining 95.12%, 93.20% and 0.9416 for the best case, respectively) to the ones of leading approaches from the competition (Se: 94.24%, Sp: 77.81%, MAcc: 0.8602, in the best case), showing a clear improvement, especially in terms of specificity.

Using a NAS in this context instead of a traditional digital audio processing approach allows us not only to achieve a very good accuracy result, but also the possibility to develop a portable diagnosis device based on the system that has been described in this paper as the next step in this line of research. This device would be fully implemented in an FPGA (see Fig. 12) where a NAS, a configurable real-time segmentation and sonogram generator, and a full-custom CNN accelerator would be programmed. The input to this system would be generated by a PDM microphone that would be placed on each of the four main auscultatory areas: Aortic area, Pulmonic area, Tricuspid area, Mitral Area (Apex). The PDM microphone directly transmits the audio signal information in a spike-based codification, which would feed the NAS' input. The fact that this device uses a NAS to decompose the audio into frequency bands instead

of using a Fourier Transform leads to having a lower power consumption. As it is presented in [44], a low-power radix-2 FFT accelerator for FPGA achieves a power consumption of 125 mW; however, the NAS' is only 29.7 mW [22], which is less than 24% of the power consumption of the FFT. Additionally, the NAS could interface directly with Spiking Convolutional Neural Networks (SCNN) without the need of the segmentation of the information and the sonogram generation, processing the auditory information in a continuous way. When connected to an SCNN, the system would only need to compute and classify the input signal when spikes are being fired. This means that if there is no activity in the input, the power consumption of the device would be even less. This "neuromorphic stethoscope" would also consist of a button to start the analysis and two LEDs, which would indicate the result of the CNN's classification result in real time as either healthy subject or pathological patient.

REFERENCES

[1] M. Nichols, N. Townsend, P. Scarborough, R. Luengo-Fernandez, J. Leal, A. Gray, and M. Rayner, "European cardiovascular disease statistics 2012: European heart network. brussels," *Eur. Soci. Cardiology*, Sophia Antipolis, 2012.

[2] D. Lloyd-Jones *et al.*, "Heart disease and stroke statistics-2010 update a report from the American heart association," *Circulation*, vol. 121, no. 7, pp. e46–e215, 2010.

[3] D. Roy, J. Sargeant, J. Gray, B. Hoyt, M. Allen, and M. Fleming, "Helping family physicians improve their cardiac auscultation skills with an interactive CD-ROM," *J. Continuing Edu. Health Professions*, vol. 22, no. 3, pp. 152–159, 2002.

[4] E. Etchells, C. Bell, and K. Robb, "Does this patient have an abnormal systolic murmur?" *Jama*, vol. 277, no. 7, pp. 564–571, 1997.

[5] S. Mangione and L. Z. Nieman, "Cardiac auscultatory skills of internal medicine and family practice trainees: a comparison of diagnostic proficiency," *Jama*, vol. 278, no. 9, pp. 717–722, 1997.

[6] M. Lam *et al.*, "Factors influencing cardiac auscultation proficiency in physician trainees," *Singapore Med. J.*, vol. 46, no. 1, pp. 11–14, 2005.

[7] S. L. Strunic, F. Rios-Gutiérrez, R. Alba-Flores, G. Nordehn, and S. Bums, "Detection and classification of cardiac murmurs using segmentation techniques and artificial neural networks," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2007, pp. 397–404.

[8] K. Ejaz, G. Nordehn, R. Alba-Flores, F. Rios-Gutierrez, S. Burns, and N. Andrisevic, "A heart murmur detection system using spectrograms and artificial neural networks." in *Proc. Int. Conf. Circuits, Signals, Syst.*, 2004, pp. 374–379.

[9] F. Rios-Gutierrez, R. Alba-Flores, and S. Strunic, "Recognition and classification of cardiac murmurs using ANN and segmentation," in *Proc. 22nd Int. Conf. Electr. Commun. Comput.*, 2012, pp. 219–223.

[10] H. M. Hadi, M. Y. Mashor, M. Z. Suboh, and M. S. Mohamed, "Classification of heart sound based on S-transform and neural network," in *Proc. 10th Int. Conf. Inf. Sci. Signal Process. Appl.*, 2010, pp. 189–192.

[11] H. Hadi, M. Mashor, M. Mohamed, and K. Tat, "Classification of heart sounds using wavelets and neural networks," in *Proc. 5th Int. Conf. Electr. Eng. Comput. Sci. Autom. Control*, 2008, pp. 177–180.

[12] L. Jia, D. Song, L. Tao, and Y. Lu, "Heart sounds classification with a fuzzy neural network method with structure learning," in *Proc. Int. Symp. Neural Netw.*, 2012, pp. 130–140.

[13] M. Singh and A. Cheema, "Heart sounds classification using feature extraction of phonocardiography signal," *Int. J. Comput. Appl.*, vol. 77, no. 4, pp. 13–17, 2013.

[14] T. Leung, P. White, W. Collis, E. Brown, and A. Salmon, "Classification of heart sounds using time-frequency method and artificial neural networks," in *Proc. 22nd Annul. Int. Conf. Eng. Med. Biol. Soc.*, 2000, vol. 2, pp. 988–991.

[15] A.-L. Noponen, S. Lukkarinen, A. Angerla, and R. Sepponen, "Phonospectrographic analysis of heart murmur in children," *BMC Pediatrics*, vol. 7, no. 1, pp. 23–33, 2007.

[16] M. Markaki, I. Germanakis, and Y. Stylianou, "Automatic classification of systolic heart murmurs," in *Proc. 2013 IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 1301–1305.

[17] I. S. Perera, F. A. Muthalif, M. Selvarathnam, M. R. Liyanaarachchi, and N. D. Nanayakkara, "Automated diagnosis of cardiac abnormalities using heart sounds," in *Proc. 2013 IEEE Point-of-Care Healthcare Technol.*, 2013, pp. 252–255.

[18] C. Potes, S. Parvaneh, A. Rahman, and B. Conroy, "Ensemble of feature-based and deep learning-based classifiers for detection of abnormal heart sounds," in *Proc. Comput. Cardiol. Conf.*, 2016, pp. 621–624.

[19] M. Zabihi, A. B. Rad, S. Kiranyaz, M. Gabbouj, and A. K. Katsaggelos, "Heart sound anomaly and quality detection using ensemble of neural networks without segmentation," in *Proc. Comput. Cardiol. Conf.*, 2016, pp. 613–616.

[20] J. Amoh and K. Odame, "Deep neural networks for identifying cough sounds," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 5, pp. 1003–1011, Oct. 2016.

[21] C. N. Gupta, R. Palaniappan, S. Swaminathan, and S. M. Krishnan, "Neural network classification of homomorphic segmented heart sounds," *Appl. Soft Comput.*, vol. 7, no. 1, pp. 286–297, 2007.

[22] A. Jiménez-Fernández *et al.*, "A binaural neuromorphic auditory sensor for FPGA: A spike signal processing approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 4, pp. 804–818, Apr. 2016.

[23] The Adress-Event Representation communication protocol. [Online]. Available: https://www.ini.uzh.ch/ amw/scx/std002.pdf

[24] The Adress-Event Representation communication protocol, 1993. [Online]. Available: https://www.ini.uzh.ch/~amw/scx/std002.pdf

[25] R. Berner, T. Delbruck, A. Civit-Balcells, and A. Linares-Barranco, "A 5 Meps $100 USB2.0 address-event monitor-sequencer interface," in *Proc. 2007 IEEE Int. Symp. Circuits Syst*, 2008, pp. 2451–2454.

[26] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, ACM, Nov. 2014, pp. 675–678.

[27] A. L. Goldberger *et al.*, "Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[28] T. Iakymchuk *et al.*, "An AER handshake-less modular infrastructure PCB with x8 2.5 Gbps LVDS serial links," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2014, pp. 1556–1559.

[29] J. P. Dominguez-Morales, A. Jimenez-Fernandez, M. Dominguez-Morales, and G. Jimenez-Moreno, "NAVIS: Neuromorphic Auditory VI-Sualizer tool," *Neurocomputing*, vol. 237, pp. 418–422, 2017.

[30] R. F. Lyon and C. Mead, "An analog electronic cochlea," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 7, pp. 1119–1134, Jul. 1988.

[31] A. Jimenez-Fernandez, A. Linares-Barranco, R. Paz-Vicente, G. Jiménez, and A. Civit, "Building blocks for spikes signals processing," in *Proc. Int. Joint Conf. Neural Netw.*, 2010, pp. 1–8.

[32] M. Yang, C. H. Chien, T. Delbruck, and S. C. Liu, "A 0.5 V 55 μW 64 $\times$ 2 channel binaural silicon cochlea for event-driven stereo-audio sensing," *IEEE J. Solid-State Circuits*, vol. 51, no. 11, pp. 2554–2569, Nov. 2016.

[33] E. Cerezuela-Escudero, M. J. Dominguez-Morales, A. Jiménez-Fernández, R. Paz-Vicente, A. Linares-Barranco, and G. Jiménez-Moreno, "Spikes monitors for FPGAs, an experimental comparative study," in *Proc. Int. Work-Conf. Artif. Neural Netw.*, 2013, pp. 179–188.

[34] C. Liu *et al.*, "An open access database for the evaluation of heart sound algorithms," *Physiological Meas.*, vol. 37, no. 12, pp. 2181–2213, 2016.

[35] NAVIS Tool GitHub, 2015. [Online]. Available: https://github.com/jpdominguez/NAVIS-Tool/

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[37] Y. LeCun *et al.*, "LeNet-5, convolutional neural networks," 2015. [Online]. Available: http://yann. lecun. com/exdb/lenet.

[38] Protocol Buffers, 2008. [Online]. Available: https://developers.google.com/protocol-buffers/

[39] R. K. Sarvadevabhatla and R. V. Babu, "Freehand sketch recognition using deep features," *CoRR*, vol. abs/1502.00254, 2015. [Online]. Available: http://arxiv.org/abs/1502.00254

[40] S. Sarraf and G. Tofighi, "Deep learning-based pipeline to recognize alzheimer's disease using fMRI data," *bioRxiv*, 2016. [Online]. Available: http://www.biorxiv.org/content/early/2016/07/31/066910

[41] A. Rios-Navarro, J. P. Dominguez-Morales, R. Tapiador-Morales, M. Dominguez-Morales, A. Jimenez-Fernandez, and A. Linares-Barranco, "A sensor fusion horse gait classification by a spiking neural network on SpiNNaker," in *Proc. Int. Conf. Artif. Neural Netw*, 2016, pp. 36–44.

[42] G. D. Clifford *et al.*, "Classification of normal/abnormal heart sound recordings: The physionet/computing in cardiology challenge 2016," in *Proc. Comput. Cardiol. Conf.*, 2016, pp. 609–612.
[43] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[44] S. Mookherjee, L. DeBrunner, and V. DeBrunner, "A low power radix-2 FFT accelerator for FPGA," in *Proc. 49th Asilomar Conf. Signals, Syst. Comput.*, 2015, pp. 447–451.

Juan P. Dominguez-Morales (M'16) received the B.S. degree in computer engineering and the M.S. degree in computer engineering and networks from the University of Seville, Sevilla, Spain, in 2014 and 2015, respectively. Since October 2015, he has been working toward the Ph.D. degree in the Computer Architecture and Technology Department, University of Seville with a research grant from the Spanish Ministry of Education and Science. His research interests include neuromorphic engineering, spiking neural networks, and neuromorphic sensors. Since January 2016, he has been a member of the IEEE Circuits and Systems Society, the IEEE Brain Community, the IEEE Signal Processing Society, and the European Neural Network Society.

Angel F. Jimenez-Fernandez received the B.S. degree in computer engineering, the M.S. degree in industrial computer science, and the Ph.D. degree in neuromorphic engineering from the University of Seville, Sevilla, Spain, in 2005, 2007, and 2010, respectively. Since October 2007, he has been an Assistant Professor of computer architecture and technology at the University of Seville. In April 2011, he was promoted to an Associate Professor. His research interests include neuromorphic engineering applied to robotics, real-time spikes signal processing, neuromorphic sensors, field programmable gate array digital design, embedded systems development, high-speed serial communications, and smart sensors networking.

Manuel J. Dominguez-Morales received the B.S. degree in computer science from the University of Seville, Sevilla, Spain, in 2008, the M.S. degree in software engineering and the M.S. degree in computer engineering, both from the University of Seville, in 2009 and 2014, respectively, and the Ph.D. degree in industrial informatics from the same University in 2014. From January 2009 to June 2014, he was an Assistant Professor at the Department of Computer Architecture and Technology, University of Seville. Since July 2014, he is a Postdoc Researcher and a Lecturer in the same department. His research fields are focused on computer vision, image processing, signal processing, programmable hardware digital design, computer architecture, robotics, and e-Health.

Gabriel Jimenez-Moreno received the M.S. degree in physics (electronics) and the Ph.D. degree from the University of Seville, Seville, Spain, in 1987 and 1992, respectively. After working with Alcatel, he was granted a Fellowship from the Spanish Science and Technology Commission (CICYT). Currently, he is an Associate Professor of computer architecture at the University of Seville. From 1996 until 1998, he was the Vice-Dean of the E.T.S. Ingenieria Informatica, University of Seville. He participated in the creation of the Department of Computer Architecture (also at the University of Seville) and since 2013 has been its Director. He is the author of various papers and research reports on robotics, rehabilitation technology, and computer architecture. He has directed three national research projects on neuromorphic systems. His research interests include neural networks, vision processing systems, embedded systems, computer interfaces, and computer architectures.

Appendix D

Deep Spiking Neural Network model for time-variant signals classification: a real-time speech recognition approach

Authors

- Juan Pedro Dominguez-Morales
- Qian Liu
- Robert James
- Daniel Gutierrez-Galan
- Angel F. Jimenez-Fernandez
- Simon Davidson
- Steve Furber

Publication

Title: Artificial Neural Networks and Machine Learning

Type: Conference Paper

Conference Name International Joint Conference on Neural Networks (IJCNN 2018)

Place Rio de Janeiro, Brazil. **Date:** July 2018

Publisher: IEEE

Pages: 4288-4295

ISBN: 978-1-5090-6014-6

Deep Spiking Neural Network model for time-variant signals classification: a real-time speech recognition approach

Juan P. Dominguez-Morales[1], Qian Liu[2], Robert James[2], Daniel Gutierrez-Galan[1],
Angel Jimenez-Fernandez[1], Simon Davidson[2], and Steve Furber[2]
[1]Robotics and Computer Technology Lab.
University of Seville, Seville, Spain
Email: jpdominguez@atc.us.es
[2]Advanced Processor Technologies Group, School of Computer Science.
University of Manchester. Manchester, UK

***Abstract*—Speech recognition has become an important task to improve the human-machine interface. Taking into account the limitations of current automatic speech recognition systems, like non-real time cloud-based solutions or power demand, recent interest for neural networks and bio-inspired systems has motivated the implementation of new techniques.**

Among them, a combination of spiking neural networks and neuromorphic auditory sensors offer an alternative to carry out the human-like speech processing task. In this approach, a spiking convolutional neural network model was implemented, in which the weights of connections were calculated by training a convolutional neural network with specific activation functions, using firing rate-based static images with the spiking information obtained from a neuromorphic cochlea.

The system was trained and tested with a large dataset that contains "left" and "right" speech commands, achieving 89.90% accuracy. A novel spiking neural network model has been proposed to adapt the network that has been trained with static images to a non-static processing approach, making it possible to classify audio signals and time series in real time.

***Index Terms*—speech recognition, audio processing, Spiking Neural Networks, Convolutional Neural Networks, neuromorphic hardware, deep learning.**

I. Introduction

Voice commands are commonly used in multiple personal virtual assistants [1], like Cortana in Microsoft Windows, or Siri in iOS. Users are able to control their personal computers or mobile phones by using natural language sentences, like "Remind me to call Robert in the afternoon", or more directly, "Call Robert". This kind of assistants are based on a field of Artificial Intelligence (AI) called Natural Language Processing (NLP) to identify what the user is saying [2], [3]. The audio is processed and analyzed using Digital Signal Processing (DSP) techniques, such as speech processing [4].

Speech recognition is the interdisciplinary sub-field of speech processing, in which spoken sentences are recognized and translated to text (or other data representation) using specific methodologies. Typically, these methods identify each spoken word in isolation, applying several processing steps to obtain features that are then mapped to a specific word [5].

In recent years, the application of Artificial Neural Network (ANN) to this field has become commonplace. Notably, the combination of Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) has led to significant progress in developing human-machine interface, as in [6], [7], [8], [9]. Recently, the Google WaveNet system [10] demonstrated significantly improved comprehension of entire conversations as well as being able to generate human-like speech from text, based on a CNN trained on raw audio voice characterization.

Training CNNs is a relatively easy task. There exist several frameworks and training mechanisms to achieve this. The most used training algorithm (for ANN and CNN training) is the well-known Levenberg-Marquardt back-propagation algorithm [11]. In contrast, there is no established standard training algorithm for Spiking Neural Networks.

Spike-Time-Dependant Plasticity (STDP) is a biological process that is able to adjust the strength (weights) of the connections between neurons based on the relative timing of a particular neuron's output and input spiking activity. This process has been implemented in several simulators and hardware platforms, including SpiNNaker [12], and has become one of the most ubiquitous approaches for training spike-based networks especially for unsupervised learning [13]. STDP has proved to be very useful and robust for static input signals like images [14], [15], but it is more difficult to apply when it comes to processing time-varying signals such as audio samples.

As an alternative to STDP, the weights of the connections between neurons in a network could be set by hand or based on particular statistical algorithms. This approach was taken into account in papers like [16], in which the authors set the weights using two different firing-rate based normalizations for classifying between eight different pure tones. This option is complex because it generally needs several trial-and-error loops in order to find the best weight configuration, which can take a long time. Also, this way of setting the weights of the connections is too task specific and lacks the generality and

978-1-5090-6014-6/18/$31.00 ©2018 IEEE

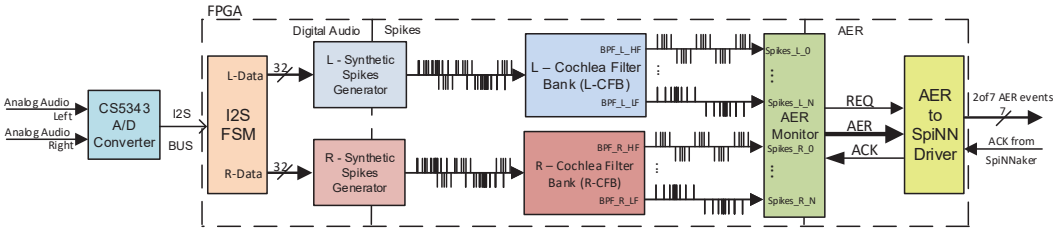


Fig. 1: Neuromorphic Auditory Sensor (NAS) block diagram.

biological plausibility of STDP.

Due to the increasing interest in SNNs, numerous works have tried to develop new frameworks or methods to automatically train SNN models. The first approach is to develop new STDP-based algorithms, as in [17], who used the force firing technique for incremental learning, making it possible to learn new patterns continually in real-time using an unsupervised learning procedure. Also, in [18], the authors used a new learning rule, named fatiguing STDP, which combines the long-term STDP dynamics with a mechanism of short-term synaptic fatigue dynamics.

There are many other bio-inspired techniques for training neural models, such as the use of evolutionary algorithms [19] to adjust the weights of the network.

In recent time, the difference in classification error between deep SNNs and deep ANNs has diminished significantly [20]. These exciting results suggest that, if trained appropriately, an SNN can be used for machine learning inference without introducing penalties in data classification accuracy. Using a deep SNN instead of a deep ANN alternative can provide a machine learning system with power saving and input noise tolerance benefits [21].

Additionally, such deep SNNs can be trained on input data generated from a neuromorphic spiking sensor device, unlocking the potential for a real-time inference system on a spiking neuromorphic platform [22]. We believe that only when these are combined the true strengths of a fully spike-based processing system will be apparent.

The aforementioned developments in deep SNNs show accurate classification of static input data (images) using a deep convolutional SNN. We show in this work that we are able to train a similarly structured network on time series input data from a Neuromorphic Auditory Sensor (NAS) [23] produced from a range of sound inputs. By using a technique of generating a training dataset consisting of many overlapping 'snapshots' of the NAS output and a 'time-buffering' input to the SNN, we are able to produce a robust inference on time varying spiking inputs.

The rest of the paper is structured as follows: section II presents an overview of the system architecture and the speech commands database that was used in this work along with how the train and test datasets were generated. Then, section III describes the whole framework that was used to train and simulate the SNN with the audio samples dataset that was obtained from the previous section. Then, section IV describes the results of both the training and the simulation. Up to this point, this setup is used for training and testing the system with static inputs (audio samples are converted into images), so in section V we propose a novel SNN architecture to use the network that was previously trained with static data in real time using a live input from a neuromorphic cochlea. Finally, the conclusions of this work are presented in section VI.

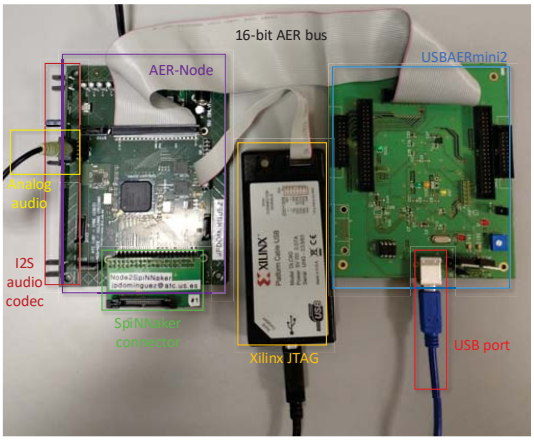
II. System overview, dataset acquisition and preprocessing of the information

In audio processing, a Digital Signal Processor is usually used to carry out large audio processing tasks, due to the computational capabilities of these devices. The neuromorphic approach uses bio-inspired devices that mimic the behavior of biological senses, reproducing with greater fidelity the individual steps by which the ear and the auditory cortex interact to process aural information.

In recent years, several researchers have developed theoretical cochlea models, using either analog or digital circuits to implement their models. As a result, many neuromorphic hardware platforms have appeared and are being used in research projects. There exist several models of analog [24] [25] [26] and digital cochleae [27] [28] [29] [30].

In this work we use a Neuromorphic Auditory Sensor [23] (NAS), which is a digital cochlea implementation. It is a FPGA-based sensor, in which all processing modules are spike-based. As it is implemented on a reconfigurable platform, this sensor's configuration parameters are flexible and can be adapted to any application.

This kind of sensors mimic how the biological cochlea processes audio signals. The cochlea is able to decompose the input audio signal into different frequency bands (also called channels). This decomposition is carried out by a series of cascade-connected stages that subtract the information from consecutive spike-based low-pass filters' output spikes in order to reject out-of-band frequencies, obtaining a response equivalent to that of a bandpass filter [23]. The entire NAS architecture is shown in Fig. 1. A flow of spikes coded as AER (Address-Event Representation) [31] events is obtained in the output, which can be either sent to the SpiNNaker board



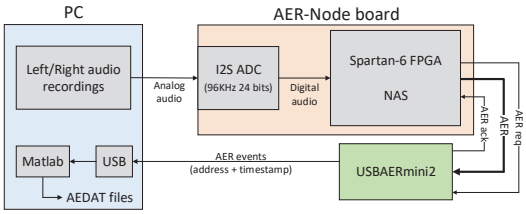


Fig. 2: Picture (top) and block diagram (bottom) of the hardware setup for the dataset generation.

through the AER-SpiNNaker interface module [32] or to the computer using a USBAERmini2 board [33].

A 32-channel mono-aural NAS was used in this work, employing this neuromorphic approach in a speech recognition task where spoken commands corresponding to the words ”left” and ”right” are classified. The Speech Command dataset, which consists of 65000 one-second long utterances of 30 short words, was used in this work. This data was collected by Google and released under a Creative Commons BY 4.0 license. Only the ”left” and ”right” voice commands of the dataset (a total of 4720 audio files from thousands of different speakers) were used in this work, since one of the final goals of the COFNET project is to drive a robot by using only these two voice commands.

Each of the audio samples were sent to the audio input of an AER-Node platform [34], which consists of a Spartan-6 FPGA in which a 32-channel mono-aural NAS is programmed. With this sensor the audio signal is decomposed into frequency bands and then packetized using the Address-Event Representation protocol (AER). An USBAERmini2 board receives this information and sends it to the computer through the USB port. A MATLAB script is used to collect the AER packets that are received through the serial port and to store them into AEDAT[1] files (one file is generated per audio sample), which is a common format used for storing this kind of information. The hardware setup used for generating the dataset is shown in Fig. 2.

These AEDAT files were then converted to sonogram images using NAVIS’s algorithms [35] in order to train a CNN. To do this, a bin width of 20 ms was selected in order to calculate the firing rate for each of the NAS’ channels in every bin. This was done by counting the number of spikes fired in that portion of time and dividing that value by 20 ms (see Algorithm 1), which is the length that was selected for this work. Fig. 3 shows images from both the ”left” and the ”right” classes after this process was carried out. In order to make the training of the network more robust to a real scenario, in which the core information of the audio could be presented not only in the center of the image but in any position of it, an overlapping shifting window was used, generating several images for each audio sample with the information centered in different timestamps.

Algorithm 1 Sonogram calculation

```
bin_width = 20 ms
sonogram = zeros(max(in_addr), max(in_timeStamp)/bin_width)
for i=1:max(in_addr) do
    sonogram(in_addr(i), in_tStamp(i)/bin_width)++
end for
sonogram = sonogram/bin_width
```

A total of 141726 images were generated in this process, 121565 of which were used to train the network and the remaining 20161 images to test it and obtain the accuracy ratio of the system.

III. Off-line SNN training and SNN construction

The general off-line SNN training method proposed by Liu et. al. [36] is based on two novel activation functions. One is Noisy Softplus (NSP) [20], which closely mimics the LIF firing activity driven by current influx with different noise levels. The other, Parametric Activation Function (PAF), maps abstract numerical numbers of activation functions to specific physical units of a spiking neuron. Thus, the combination provides an equivalent representation of a spiking LIF neuron with abstract activation functions of ANNs. PAF allows using more generalized activation functions (e.g., ReLU instead of NSP) to model a LIF neuron once its parameters are fitted by NSP. Therefore, the weights of a SNN can be trained off-line on an equivalent ANN exactly the same way as conventional ANNs (e.g., using Backpropagation and Stochastic Gradient Descent), but using PAFs. The simple steps can be described as follows: firstly, estimate the parameter of PAFs; then, train an equivalent ANN using the PAF version of the activation functions (e.g., PAF-ReLU); finally, transfer the trained weights back to the SNN without further transformation.

The off-line SNN training tool is published in Github[2]. It is comprised of two main parts: the Matlab code for ANN

[1]https://inilabs.com/support/software/fileformat
[2]https://github.com/qian-liu/off_line_SNN

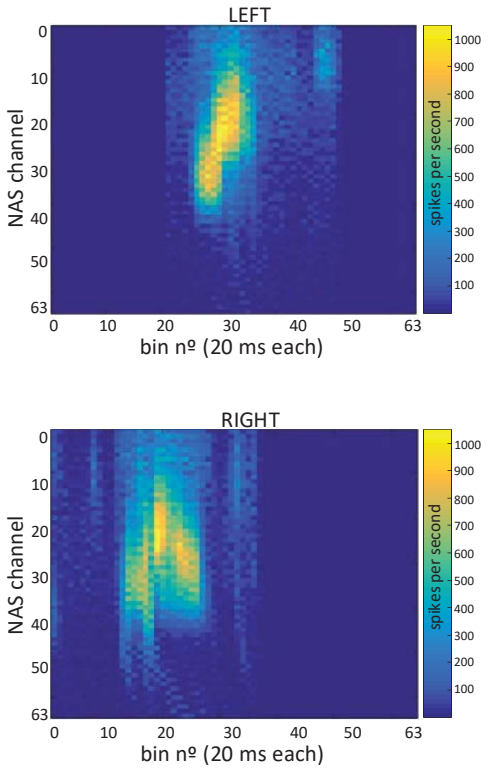


Fig. 3: Sonogram images corresponding to one "Left" (top) and one "Right" (bottom) audio samples from the Speech Command dataset after obtaining their spiking information from the NAS.

training and the Python code for reading trained weights and translating into PyNN language. The Matlab code is based on an ANN training tool called DeepLearnToolbox[3], and we implemented the two activation functions described above. It is worth noting that the NSP and its derivative takes two variables as inputs: the mean of the noisy current x and its variance σ. Therefore, the computation of both the forward and backward paths are doubled and the state to be stored is also doubled in size. The PAF is easily implemented by multiplying the parameter p of the original activation function: $p \times f(x)$.

The Python code reads the network architecture of an ANN layer-by-layer, and constructs equivalent populations of LIF neurons accordingly. It then takes the layer-wise weights of the ANN and translates them to the connection list between populations of LIF neurons. After the building-up phase, the testing code (which is simulated in NEST) generates Poisson Spike trains based on parameter configurations and the intensity of pixels of an input image; then, it feeds the network with the spike trains and records the output spike trains on the classification layer; finally, it analyses the results where the highest firing rates determine the class to which an image is assigned. The overall performance on the whole testing dataset is then compared with the ANN testing result.

[3]https://github.com/rasmusbergpalm/DeepLearnToolbox

IV. Results

A 5C-3P-3C-2P Spiking Convolutional Neural Network (5x5 kernel-size convolutional layer followed by a 2x2 pooling layer, another 3x3 convolutional layer followed by a 2x2 pooling layer, and then a fully connected layer) was trained in Matlab with rate-based sonograms (See Fig. 3) that contained the firing rate information obtained from a NAS using "left" and "right" speech commands from a well-known open database that was presented in section II. The CNN architecture is shown in Fig. 4.

An accuracy result of 92.21% was achieved when training the CNN in Matlab for 30 epochs, at a learning rate value of 0.1 and a synaptic time constant of 0.005 ms, using the ReLU activation function on the fully connected layer. After this, the network was fine-tuned for one more epoch with the Noisy Softplus activation function that was described in section III, starting off with the weights of the connections that were obtained from the previous step (using ReLU as the activation function).

After the fine-tuning process, the performance of the network was almost the same, obtaining 90.80% accuracy. As was explained in previous sections, the trained weights were tweaked (fine-tuned) in this process, resulting on a slightly lower accuracy value in this case (less than 2% decrease), but improving the performance when translating from the ANN in Matlab to a SNN in pyNN (NEST).

The weights obtained from the ANN training and fine-tuning in Matlab were then saved and used to test a SNN. The SNN was built in pyNN for the NEST simulator based on the architecture of the ANN that was trained in the previous step. The network was tested using 20161 samples, achieving 89.90% accuracy (the confusion matrix is shown in Fig. 5). As can be observed, the result obtained in the "left"/"right" classification in the SNN simulation that was run on NEST is almost the same as the one that was obtained when training the ANN in Matlab, meaning that, with this process, the authors have found a proper way to train audio signals (or time series) without compromising the accuracy of the network.

Tests were carried out using the NEST simulator and also deploying the whole SNN model in a 48-chip SpiNNaker hardware platform. In future works, the authors would be focusing on making use of the NAS-SpiNNaker live connection [37] to test the speech commands recognition using a real-time input from a microphone connected to the NAS. The next section will describe the SNN architecture for testing this approach in real-time.

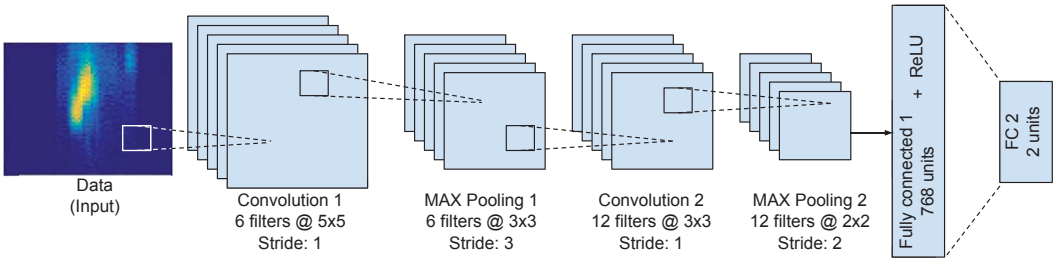


Fig. 4: CNN architecture used for training the ”left”/”right” commands in Matlab.

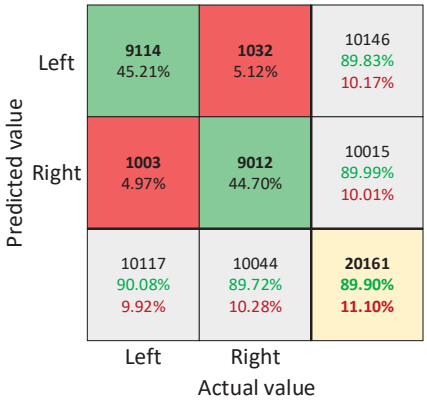


Fig. 5: Confusion matrix of the SNN test using 20161 samples (10117 ”left” and 10044 ”right” samples).

V. SNN ARCHITECTURE FOR AUDIO SAMPLES CLASSIFICATION IN REAL TIME

The accuracy result of the system achieved when using the method described in section III proves that this mechanism could be used to classify audio samples like the ones used in this work or even more complex ones as long as they can be converted into images.

This is a completely offline approach, which means that audio samples are not being inputted in real time. These samples have to be already recorded and converted into spikes in order to classify them. The point on using the NAS is that, besides of processing the sound information in a bio-inspired way, it is able to provide a real time output with the audio signal decomposed into frequency bands (32 bands or cochlea channels in this case) and already converted into spikes, as the biological cochlea would do.

Even when not making use of the real-time capabilities of this neuromorphic sensor, using it could be useful for tasks in which the classification does not need to be done in a short period of time. In [38], Dominguez-Morales et al. use a NAS to process heart sounds recordings and classify whether it is a healthy person or a pathological patient in order to help cardiologists in the auscultation process. Applications like this do not require an immediate output from the classifier, meaning that the sound could be recorded and analyzed later.

However, in tasks like robot navigation with speech commands, recognizing the command and acting on the motors of the robot are actions that need to be done as soon as possible. Otherwise, the navigation would not feel fluid and natural. One of the main goals of the COFNET project (TEC2016-77785-P) is to drive a 4-wheel SUMMIT XL robot from Robotnik using the fusion of the neuromorphic information coming from a neuromorphic retina (Dynamic Vision Sensor) and from a NAS (using speech commands). To accomplish this while using the same training approach considered in this work, the authors propose the SNN architecture shown in Fig. 6.

This architecture takes into account that the input data has been trained using a deep Spiking Convolutional Neural Network (SCNN) as it was a static input (image). That is, the image is converted from a matrix (two-dimensional array) to a single dimension array by flattening the matrix (e.g. a 28x28 MNIST image is converted to an array of 768 elements). The whole trained SCNN is presented in the figure in a cloud shape. To adapt the trained model in order to use real-time input from the NAS, a new layer of spiking populations has to be added.

These populations act as a layer between the NAS and the trained SCNN and its goal is to adapt the spiking information that comes out of the NAS in real time in order to serve as input to the network. This is done by having 64 populations (due to the fact that the network is trained with 64 20 ms-bins images) of 64 neurons each (two neurons per NAS channel) that are connected like a daisy chain, with delayed one-to-one connections between every two. The delay that is set for these connections is 20 ms, because of the bin width used. NAS's output is connected to the first of these populations, propagating the same spiking information to the next one after 20 ms. Then, each of the 64 populations of this layer is connected to the previously trained SCNN with no delay. This way, as soon as the speech command is sent to the NAS, the populations between the NAS and the SCNN will start to

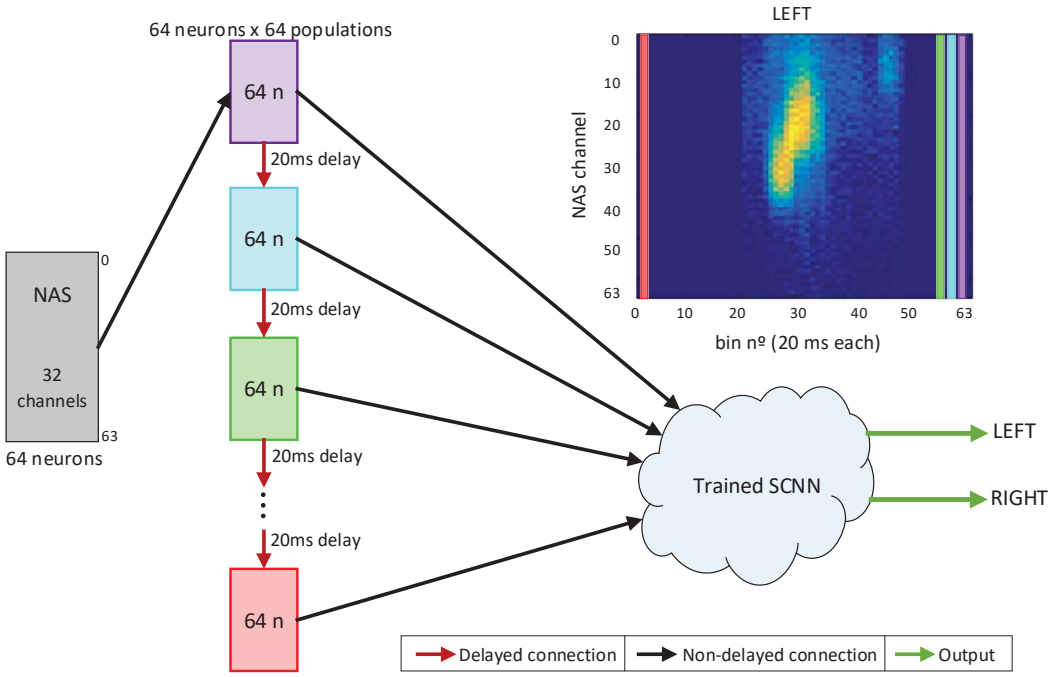


Fig. 6: Real-time NAS audio input SCNN scenario with a buffering layer consisting of a set of delayed populations.

propagate the information. This ”time-buffered” architecture allows to run real-time experiments for speech recognition and audio samples classification with a previous step of training the network with static audio images, which is a novelty in the neuromorphic engineering field. Having several images for the same speech command with the information shifted and centered in a different bin allows not only the training to be more robust but also the real-time test to take less time to start providing the correct result. That is, spikes from the NAS do not need to propagate through many populations to be classified correctly since the network was trained to recognize that the important information of the speech command could also appear in the first bins (which correspond to the first populations) instead of just in the middle section of the sonogram.

VI. Conclusions

In this work, the authors have presented a novel mechanism for training time series offline and testing them later in real time in a Spiking Convolutional Neural Network with the information obtained from the live output of a Neuromorphic Auditory Sensor.

The results obtained in this work prove that almost the same accuracy results (1% less in this case) can be achieved when testing a deep Spiking Neural Network using the weights obtained from a previously trained Convolutional Neural Network with spike-rate based images, which is a novelty for time-dependent signals like audio signals.

A database with 4720 ”left” and ”right” speech commands from the Speech Commands Dataset was used to generate 141726 sonogram images with the spiking information obtained from a neuromorphic cochlea (NAS). These images were later used for training and testing the system, achieving an accuracy result of 89.90% when simulating and deploying the network in the SpiNNaker hardware platform.

The authors have also presented a novel SNN architecture for audio samples classification in real time using the output from a neuromorphic sensor as input to the network and a buffering layer with delayed populations that adapts the information from a real-time domain to a static domain, in which the SNN is trained for. This approach could also be used for processing time series or time-dependent signals with SNNs in real time.

Acknowledgment

This work is supported by the Spanish government grant (with support from the European Regional Development Fund) COFNET (TEC2016-77785-P). The work of Juan P. Dominguez-Morales was supported by a Formación de Personal Universitario Scholarship from the Spanish Ministry of

Education, Culture and Sport. Juan P. Dominguez-Morales and Daniel Gutierrez-Galan would like to thank Simon Davidson, Steve Furber and the whole APT group for their kindness and constant help during their stay in Manchester.

REFERENCES

[1] R. S. Cooper, J. F. McElroy, W. Rolandi, D. Sanders, R. M. Ulmer, and E. Peebles, "Personal virtual assistant," Jun. 29 2004, uS Patent 6,757,362.

[2] G. G. Chowdhury, "Natural language processing," *Annual review of information science and technology*, vol. 37, no. 1, pp. 51–89, 2003.

[3] G. Gazdar and C. S. Mellish, *Natural language processing in Lisp: An introduction to computational linguistics*. Addison-Wesley Wokingham, England, 1989.

[4] X. Huang, A. Acero, H.-W. Hon, and R. Reddy, *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR Upper Saddle River, 2001, vol. 95.

[5] S. Furui, "Speaker-independent isolated word recognition using dynamic features of speech spectrum," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 1, pp. 52–59, 1986.

[6] G. Williams and S. Renals, "Confidence measures for hybrid HMM/ANN speech recognition." 1997.

[7] P. McGuire, J. Fritsch, J. J. Steil, F. Rothling, G. A. Fink, S. Wachsmuth, G. Sagerer, and H. Ritter, "Multi-modal human-machine communication for instructing robot grasping tasks," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2. IEEE, 2002, pp. 1082–1088.

[8] O. Russakovsky, L.-J. Li, and L. Fei-Fei, "Best of both worlds: human-machine collaboration for object annotation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2121–2131.

[9] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[10] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[11] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[12] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, 2013.

[13] P. U. Diehl and M. Cook, "Efficient implementation of STDP rules on SpiNNaker neuromorphic hardware," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 4288–4295.

[14] T. Iakymchuk, A. Rosado-Muñoz, J. F. Guerrero-Martínez, M. Bataller-Mompeán, and J. V. Francés-Víllora, "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, p. 4, 2015.

[15] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep neural networks for object recognition," *arXiv preprint arXiv:1611.01421*, 2016.

[16] J. P. Dominguez-Morales, A. Jimenez-Fernandez, A. Rios-Navarro, E. Cerezuela-Escudero, D. Gutierrez-Galan, M. J. Dominguez-Morales, and G. Jimenez-Moreno, "Multilayer spiking neural network for audio samples classification using SpiNNaker," in *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 45–53.

[17] Z. Hu, T. Wang, and X. Hu, "An STDP-based supervised learning algorithm for spiking neural networks," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 92–100.

[18] T. Moraitis, A. Sebastian, I. Boybat, M. Le Gallo, T. Tuma, and E. Eleftheriou, "Fatiguing STDP: Learning from spike-timing codes in the presence of rate codes," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 1823–1830.

[19] N. Pavlidis, O. Tasoulis, V. P. Plagianakos, G. Nikiforidis, and M. Vrahatis, "Spiking neural network training using evolutionary algorithms," in *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 4. IEEE, 2005, pp. 2190–2194.

[20] Q. Liu and S. Furber, "Noisy Softplus: A biology inspired activation function," in *International Conference on Neural Information Processing*. Springer, 2016, pp. 405–412.

[21] C. Farabet, R. Paz, J. Perez-Carrasco, C. Zamarreo, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, and B. Linares-Barranco, "Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing," *Frontiers in Neuroscience*, vol. 6, p. 32, 2012. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2012.00032

[22] F. Perez-Peña, J. A. Leñero-Bardallo, A. Linares-Barranco, and E. Chicca, "Towards bioinspired close-loop local motor control: A simulated approach supporting neuromorphic implementations," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–4.

[23] A. Jiménez-Fernández, E. Cerezuela-Escudero, L. Miró-Amarante, M. J. Domínguez-Morales, F. de Asís Gómez-Rodríguez, A. Linares-Barranco, and G. Jiménez-Moreno, "A binaural neuromorphic auditory sensor for FPGA: a spike signal processing approach," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 4, pp. 804–818, 2017.

[24] S.-C. Liu, A. Van Schaik, B. A. Mincti, and T. Delbruck, "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 2027–2030.

[25] B. Wen and K. Boahen, "A silicon cochlea with active coupling," *IEEE transactions on biomedical circuits and systems*, vol. 3, no. 6, pp. 444–455, 2009.

[26] T. J. Hamilton, C. Jin, A. van Schaik, and J. Tapson, "An active 2-D silicon cochlea," *IEEE Transactions on biomedical circuits and systems*, vol. 2, no. 1, pp. 30–43, 2008.

[27] C. D. Summerfield and R. F. Lyon, "ASIC implementation of the Lyon cochlea model," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 5. IEEE, 1992, pp. 673–676.

[28] A. Mishra and A. E. Hubbard, "A cochlear filter implemented with a field-programmable gate array," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 1, pp. 54–60, 2002.

[29] M.-P. Leong, C. T. Jin, and P. H. Leong, "An FPGA-based electronic cochlea," *EURASIP Journal on Applied Signal Processing*, vol. 2003, pp. 629–638, 2003.

[30] I. Gambin, I. Grech, O. Casha, E. Gatt, and J. Micallef, "Digital cochlea model implementation using Xilinx XC3S500E spartan-3E FPGA," in *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 946–949.

[31] The Adress-Event Representation communication protocol. [Online]. Available: https://www.ini.uzh.ch/ amw/scx/std002.pdf

[32] L. Plana, J. Heathcote, J. Pepper, S. Davidson, J. Garside, S. Temple, and S. Furber, "spI/O: A library of FPGA designs and reusable modules for I/O in SpiNNaker systems."

[33] R. Berner, T. Delbruck, A. Civit-Balcells, and A. Linares-Barranco, "A 5 Meps $100 USB2.0 address-event monitor-sequencer interface," in *2007 IEEE International Symposium on Circuits and Systems*. IEEE, 2007, pp. 2451–2454.

[34] T. Iakymchuk, A. Rosado, T. Serrano-Gotarredona, B. Linares-Barranco, A. Jimenez-Fernandez, A. Linares-Barranco, and G. Jimenez-Moreno, "An AER handshake-less modular infrastructure PCB with x8 2.5 Gbps LVDS serial links," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 1556–1559.

[35] J. P. Dominguez-Morales, A. Jimenez-Fernandez, M. Dominguez-Morales, and G. Jimenez-Moreno, "NAVIS: Neuromorphic Auditory VISualizer tool," *Neurocomputing*, vol. 237, pp. 418–422, 2017.

[36] Q. Liu, Y. Chen, and S. Furber, "Noisy Softplus: an activation function that enables SNNs to be trained as ANNs," *arXiv preprint arXiv:1706.03609*, 2017.

[37] J. P. Dominguez-Morales, A. Rios-Navarro, D. Gutierrez-Galan, R. Tapiador-Morales, A. Jimenez-Fernandez, E. Cerezuela-Escudero, M. Dominguez-Morales, and A. Linares-Barranco, "Live demonstration-multilayer spiking neural network for audio samples classification using SpiNNaker," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–1.

[38] J. P. Dominguez-Morales, A. F. Jimenez-Fernandez, M. J. Dominguez-Morales, and G. Jimenez-Moreno, "Deep neural networks for the recognition and classification of heart murmurs using neuromorphic au-

ditory sensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 24–34, Feb 2018.